TUGAS AKHIR – SS141501

KLASIFIKASI GEN YANG TERKAIT SINDROM ALZHEIMER MENGGUNAKAN METODE *NAÏVE BAYES CLASSIFIER, BINARY LOGISTIC REGRESSION* DAN *LOGISTIC REGRESSION ENSEMBLE*

REYNALDI WISNU WERDHANA
NRP 1313 100 097

Dosen Pembimbing
Dr.rer.pol. Heri Kuswanto

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

TUGAS AKHIR – SS141501

KLASIFIKASI GEN YANG TERKAIT SINDROM ALZHEIMER MENGGUNAKAN METODE *NAÏVE BAYES CLASSIFIER, BINARY LOGISTIC REGRESSION* DAN *LOGISTIC REGRESSION ENSEMBLE*

REYNALDI WISNU WERDHANA
NRP 1313 100 097

Dosen Pembimbing
Dr.rer.pol. Heri Kuswanto

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

FINAL PROJECT– SS141501

CLASSIFICATION OF ALZHEIMER'S DISEASE RELATED GENES USING NAÏVE BAYES CLASSIFIER, BINARY LOGISTIC REGRESSION AND LOGISTIC REGRESSION ENSEMBLE

REYNALDI WISNU WERDHANA
NRP 1313 100 097

Supervisors
Dr.rer.pol. Heri Kuswanto

UNDERGRADUATE PROGRAM
DEPARTMENT OF STATISTICS
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

LEMBAR PENGESAHAN

KLASIFIKASI GEN YANG TERKAIT SINDROM ALZHEIMER MENGGUNAKAN METODE *NAIVE BAYES CLASSIFIER, BINARY LOGISTIC RERGESSION* DAN *LOGISTIC REGRESSION ENSEMBLE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Sains
pada
Program Studi Sarjana Departemen Statistika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember

Oleh :
REYNALDI WISNU WERDHANA
NRP. 1313 100 097

Disetujui oleh Pembimbing:
Dr.rer.pol. Heri Kuswanto)
NIP. 19820326 200312 1 004

Mengetahui,
Kepala Departemen

Dr. Suhartono
NIP. 19710929 199512 1 001

SURABAYA, JULI 2017

KLASIFIKASI GEN YANG TERKAIT SINDROM ALZHEIMER MENGGUNAKAN METODE *NAÏVE BAYES CLASSIFIER* DAN *LOGISTIC REGRESSION ENSEMBLE*

Nama	**: Reynaldi Wisnu Werdhana**
NRP	**: 1313 100 097**
Departemen	**: Statistika**
Pembimbing	**: Dr.rer.pol. Heri Kuswanto, M.Si.**

Abstrak

Alzheimer merupakan penyakit degeneratif dan penyebab paling umum dari kasus dimensia. Salah satu kunci menangani penyakit ini adalah deteksi dini. Deteksi tersebut dapat diketahui melalui ekspresi dari gen yang terkandung dalam DNA, dengan memanfaatkan teknologi Microarray DNA. Masalah paling mendasar dalam memprediksi ekspresi adalah mendapatkan metode terbaik. Dalam penelitian ini, metode Logistic Regression Ensemble (LORENS) akan dibandingkan dengan metode Naive Bayes Classifier serta Binary Logistic Regression dengan mempertimbangkan 20 variabel yang diduga berpengaruh dalam proses klasifikasi. Variabel dalam penelitian ini berjumlah 178, yang terdiri dari 2 kelas yaitu gen Alzheimer sebanyak 98 pengamatan dan gen normal sebanyak 80 pengamatan. Hasil analisis menggunakan prosedur evaluasi full training set menghasilkan metode terbaik adalah metode LORENS 4 partisi dan threshold 0,5 memberikan hasil paling baik. Akurasi yang dihasilkan model ini adalah 76,4% dan nilai AUC 0,774. Dengan menggunakan prosedur evaluasi Cross Validation, metode LORENS adalah metode terbaik. Metode LORENS dengan 10 folds memberikan hasil partisi optimal yang digunakan adalah 5 partisi dengan threshold 0,5. Akurasi yang dihasilkan sebesar 75,28% dan nilai AUC sebesar 0,759. Metode terbaik untuk menangani masalah klasifikasi gen menggunakan data microarray dalam penelitian ini adalah metode LORENS Cross Validation 5 partisi dengan threshold 0,5.

Kata Kunci: Alzheimer, Microarray, LORENS, Naive Bayes Classifier, Cross Validation, AUC

(Halaman ini sengaja dikosongkan)

CLASSIFICATION OF ALZHEIMER'S DISEASE RELATED GENES USING NAÏVE BAYES CLASSIFIER, BINARY LOGISTIC REGRESSION AND LOGISTIC REGRESSION ENSEMBLE

Student's Name	**: Reynaldi Wisnu Werdhana**
NRP	**: 1313 100 097**
Departement	**: Statistics**
Supervisor	**: Dr.rer.pol. Heri Kuswanto, M.Si.**

Abstrak

Alzheimer is a degenerative disease and most common case of dementia. One of the keys to treat Alzheimer is early detection. The detection can be carned out by analyzing the expression of the genes contained in DNA, using DNA microarray technology. The most basic problem in classification is to find a best method. In this research, Logistic Regression Ensemble (LORENS) is applied and compared with Naïve Bayes Classifier and Binary Logistic Regression. Research examines to 178 observation, consisting of 2 classes, where 98 observations as a Alzheimer's genes and 80 observations as a normal genes. The result of the analysis using full training set found that LORENS with 4 partitions and threshold of 0,5 is the best setting. This method has accuracy of is the best method. Meanwhile, LORENS has been proven to outier from the others by Cross Validation evalution, where the optional result is obtained by 5 partition and threshold of 0,5. The accuracy is 75,28% with AUC of 0,759.

Keywords: *Alzheimer, Microarray, LORENS, Naive Bayes Classifier, Cross Validation, AUC*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Assalamu'alaikum Warahmatullah Wabarokatuh.

Puji syukur alhamdulillah senantiasa penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul

"KLASIFIKASI GEN YANG TERKAIT SINDROM ALZHEIMER MENGGUNAKAN METODE *NAÏVE BAYES CLASSIFIER, BINARY LOGISTIC REGRESSSION* DAN *LOGISTIC REGRESSION ENSEMBLE*"

Sholawat dan salam tak lupa penulis sampaikan pada junjungan besar Nabi Muhammad SAW. Dalam menyelesaikan laporan Tugas Akhir ini penulis telah banyak menerima bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis mengucapkan terima kasih kepada :

1. Dr. rer. pol. Heri Kuswanto selaku dosen pembimbing, yang telah membimbing saya, memberikan segala masukan, waktu serta pengetahuan demi terselesaikannya Tugas Akhir ini.
2. Dr. Suhartono selaku Ketua Departemen Statistika ITS yang telah memberikan fasilitas dan sarana dalam penyusunan Tugas Akhir ini.
3. Dr. Suhartono dan Ibu Santi Wulan Purnami, Ph.d. selaku dosen penguji, yang telah memberikan banyak saran, kritik dan masukan demi kesempurnaan Tugas Akhir saya.
4. Dr. Sutikno, M.Si selaku Ketua Program Studi S1 Statistika dan segenap dosen maupun tenaga pendidik Departemen Statistika ITS.
5. Kedua orang tua tercinta dan keluarga besar yang telah melimpahkan kasih sayang dan segala doa.
6. Sahabat-sahabat dari SMA sampai sekarang, sahabat yang dipertemukan waktu kuliah "GNG" yang selalu

memberikan motivasi dan bantuan dalam bentuk apapun kepada penulis.

7. Teman-teman S1 Statistika angkatan 2013 yang berjuang bersama dalam penyelesaian Tugas Akhir, terima kasih atas dukungan dan segala bantuan dalam penyelesaian Tugas Akhir.
8. Semua pihak yang memberikan semangat serta motivasi kepada penulis untuk terus menggapai cita-cita.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang bersifat membangun sangat diharapkan.

Wassalamu'alaikum Warahmatullah Wabarokatuh.

Surabaya, Juli 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 LatarBelakang

Otak merupakan pusat sistem saraf yang mengatur seluruh kegiatan didalam tubuh makhluk hidup. Gangguan atau penyakit sekecil apapun pada otak dapat mempengaruhi kegiatan yang terjadi didalam tubuh makhluk hidup. Salah satu penyakit yang menyerang otak manusia adalah penyakit Alzheimer. Alzheimer pertama kali diidentifikasi lebih dari 100 tahun yang lalu. Penyakit ini merupakan penyakit degeneratif dan penyebab paling umum dari kasus dimensia. Hal ini ditandai dengan penurunan memori/ingatan manusia, penurunan kemampuan memecahakan masalah dan ketrampilan kognitif yang lainnya. Akibatnya kemampuan seseorang untuk melakukan aktivitas sehari-hari akan terganggu atau bahkan tidak bisa melakukan aktivitas sama sekali dalam kondisi dimensia yang parah. Penurunan kemampuan ini terjadi karena sel-sel syaraf (neuron) di bagian otak yang terlibat dalam fungsi kognitif telah rusak dan biasanya tidak berfungsi lagi (Anonim, 2016). Meskipun banyak penelitian yang meneliti tentang penyaki ini, masih banyak hal yang belum terungkap mengenai penyakit ini. Terutama tentang perubahan biologis yang menyebabkan terjadinya Alzheimer, mengapa penyakit ini dapat berlangsung lebih cepat pada beberapa orang dan bagaimana penyakit ini bisa dicegah atau bahkan dihentikan. Para peneliti percaya bahwa salah satu kunci menangani penyakit ini adalah deteksi dini. Deteksi tersebut dapat diketahui dengan melihat ekspresi dari gen yang terkandung dalam DNA.

Salah satu komponen didalam tubuh makhluk hidup yang memuat informasi penting tentang makhluk hidup tersebut adalah DNA (*Deoxyribose-Nucleic Acid*). DNA merupakan pembawa informasi genetik dari makhluk hidup. Selain itu, DNA juga dapat digunakan untuk mengetahui penyakit ataupun sindrom yang dialami oleh makhluk hidup. Salah satu cara untuk memperoleh informasi genetik dari DNA adalah dengan cara melihat ekspresi

gen dari suatu sel atau jaringan suatu mikroorganisme. Teknologi dalam bidang biologi molekuler dan bioinformatika yang mendeteksi ekspresi gen dengan jumlah yang besar sekaligus adalah DNA Microarray. Microarray merupakan chip yang berukuran kecil dan terbuat dari lempengan kaca yang berisi ribuan gen dalam bentuk fragmen DNA. Teknik ini mampu membaca ekspresi gen dalam jaringan yang berbeda dengan baik. Aplikasi microarray DNA banyak digunakan dalam pendeteksian penyakit seperti kanker, tumor dll. Teknologi ini menggunkan sampel DNA normal dan abnormal, dimana masing-masing samoel tersebut diamplifikasi dan diberi warna fluoroscent yang berbeda-beda. Biasanya, DNA normal akan diberi warna hijau dan DNA yang abnormal akan diberi warna merah. Warna-warna tersebut akan diubah menjadi data yang layak untuk diolah secara kuantitatif. Klasifikasi dan seleksi gen dengan menggunakan data microarray pernah dilakukan oleh Moorthy dan Mohamad (2011) dengan menggunakan metode Random Forest. Pada umumnya, microarray merupakan *high dimensional data*, dimana jumlah variabel yang diamati sangat besar dan jumlah pengamatan yang dilakukan jauh lebih kecil dari jumlah variabel. Data microarray biasanya digunakan untuk mengklasifikasikan gen yang normal dan gen yang tidak normal atau terindikasi terjangkit penyakit. Penelitian yang dilakukan selama ini bertujuan untuk mendapatkan metode klasifikasi yang tepat untuk memprediksi gen-gen yang terjangkit Alzheimer dan gen yang normal. Metode klasifikasi yang umum digunakan untuk dalam klasifikasi data microarray adalah Regresi Logistik, *Support Vector Machines* (SVM), *Neural Network*, *Decision Tree* dan *Naive Bayes*. Seperti penelitian yang dilakukan oleh Ambica, Gandi, & Kothalanka (2013) yang meneliti tentang klasifikasi penyakit diabetes menggunakan metode *Naive Bayes Classifier* dan *Proposed Approach*. Penelitian tersebut menghasilkan kesimpulan bahwa metode *Naive Bayes Classifier* lebih optimal dalam melakukan klasifikasi.

Masalah yang paling mendasar dalam memprediksi ekspresi gen dengan menggunakan data microarray adalah mendapatkan metode dan model terbaik yang dapat menganalisis dengan tepat. Data microarray yang pada umumnya merupakan *high dimensional data* mengharuskan metode klasifikasi statistika sebaiknya dilakukan dengan pendekatan komputasi. Tantangan para analis sekarang ini adalah *big data* dan *high dimensional data.* Pendekatan statistik yang mengharuskan untuk memenuhi asumsi-asumsi tertentu akan menjadi lemah ketika dihadapkan dengan *big data* ataupun *high dimentional data.* Pendekatan parametrik yang menggunakan pengujian signifikansi juga akan menjadi lemah ketika dihadapkan dengan *big data* ataupun *high dimentional data.* Hal itu dikarenakan *p-value* sensitif terhadap banyaknya observasi yang dilakukan (Lin, Lucas, & Shmueli, 2013). Permasalahan yang terjadi pada tahap pengujian hipotesis dengan pendekatan statistika inferensial adalah pembuktian hipotesa *null* dapat ditolak. Pengujian dengan menggunakan data yang besar cenderung menghasilkan keputusan bahwa parameter yang diuji berpengaruh signifikan karena *p-value* cenderung bernilai 0. Pendekatan parametrik untuk menganalisa data dengan jumlah yang besar akan menjadi tidak berguna, bahkan menghasilkan kesimpulan yang salah. Pendekatan komputasional dikembangkan untuk menangani kelemahan pendekatan inferensial, karena pendekatan komputasional tidak mengenal pengujian asumsi dan pengujian parameter. Pendekatan komputasional sangat dapat dipercaya, karena pendekatan ini mengadaptasi pendekatan inferensial dan menyempurnakannya dengan algoritma yang agregatif.

Pada penelitian sebelumnya tentang klasifikasi dan seleksi gen yang terkait penyakit Alzheimer pernah dilakukan oleh Nishiwaki, Kanamori, & Ohwada (2015), dimana penelitian ini menggunakan 5 dataset microarray yang dianalisis dengan menggunakan metode *Random Forest*. Penelitian ini menyeleksi 11.555 gen menjadi 50 gen, yang kemudian dirangking berdasarkan nilai *average importance score* dan digunakan hanya

20 gen dengan rangking teratas. Penelitian ini memberikan hasil yang tepat dalam mengidentifikasi gen yang terkait penyakit Alzheimer. Penelitian dengan menggunakan data microarray juga pernah dilakukan untuk klasifikasi gen yang terkait penyakit kanker. Matsumoto, Aoki & Ohwada (2015) menggunakan metode *Random Forest* dan SVM untuk memprediksi proteksi radiasi dan toksisitas. Dalam prediksi fungsi proteksi radiasi, metode SVM menghasilkan akurasi yang lebih baik dibandingkan metode *Random Forest.* Sebaliknya, metode *Random Forest* memberikan akurasi yang lebih baik dibandingkan metode SVM saat memprediksi toksisitas.

Sebuah metode klasifikasi baru telah dikembagkan oleh Lim pada tahun 2007 dengan menggunakan algoritma *Classification by Ensembles from Random Partition* (CERP) pada metode klasifikasi regresi logistik biner. Metode baru tersebut memperbolehkan data kategori menjadi variabel prediktornya. Algoritma CERP mempartisi variabel prediktor menjadi beberapa subruang. Model-model berbasis *Logistic Regression* dari masing-masing partisi yang didapat kemudian akan digabung kembali menjadi satu fungsi. Metode tersebut dikenal dengan nama *Logistic Regression Ensembles* (LORENS). Metode LORENS memiliki keunggulan, karena menggunakan algoritma CERP yang menyebabkan variabel prediktor menjadi saling *mutually exclusive* dan dibangun dari sifat *Logistic Regression* yang informatif dan juga representatif (Lee, Ahn, Moon, Kodell, & Chen, 2013). LORENS diciptakan untuk mengatasi kasus dengan banyak variabel prediktor mempunyai jumlah yang jauh lebih besar daripada pengamatan yang dilakukan. Dalam metode klasifikasi, pada umumnya *threshold* yang digunakan adalah 0,5. Hal tersebut menjadi sebuah masalah, karena tidak adil jika probabilitas masing-masing kelas dinyatakan bernilai 0,5. LORENS mampu mengatasi masalah tersebut dengan menyediakan *threshold* yang optimal untuk masing-masing kelas.

Penelitian dengan menggunakan metode LORENS pernah dilakukan oleh Lim (2010) dalam kasus AML (*Acute Myloid*

Leukimia). Dalam penelitian tersebut, didapatkan kesimpulan bahwa metode LORENS terbukti meningkatkan akurasi, *sensitivity* dan *specificity* disbanding metode klasifikasi lainnya. Penelitian serupa juga pernah dilakukan oleh Kuswanto, Asfihani, Sarumaha, & Ohwada (2015), dimana LORENS digunakan dalam mengklasifikasikan kasus pembelotan konsumen dengan ukuran sample yang sangat besar. Kemampuan LORENS dalam menangani *big data*, ketidak-seimbangan variabel respon, dan ketimpangan variabel prediktor yang cukup baik, LORENS disimpulkan lebih terpercaya walaupun tidak bisa menjelaskan hubungan antar variabel karena tidak dapat menghasilkan model yang intepretatif. Metode LORENS juga pernah digunakan oleh Zakharov & Dupont (2011) untuk menanggani data microarray. Hasil dari penelitian tersebut menyebutkan bahwa LORENS menghasilkan hasil klasifikasi yang lebih stabil dan jauh lebik akurat daripada menggunakan regresi logistik. LORENS mampu menanggani kasus data dengan jumlah observasi yang jauh lebih sedikit daripada jumlah variabelnya.

Berdasarkan keunggulan dari LORENS yang telah dinyatakan diatas, penelitian ini sangat tepat menggunakan metode tersebut untuk memprediksi gen-gen yang terkait dengan penyakit Alzheimer. Pada kasus ini, prediksi gen yang terkait penyakit Alzheimer menggunakan *high dimensional data* sebagai sumber datanya, sehingga LORENS sangat tepat digunakan. Hasil klasifikasi dari LORENS akan dibandingkan dengan hasil klasifikasi menggunakan *Naïve Bayes Classifier* (NBC). Metode NBC memiliki algoritma yang lebih sederhana dibandingkan dengan LORENS. Dalam kasus microarray, metode NBC juga pernah digunakan untuk memprediksi penyakit jantung. Penelitian yang dilakukan oleh Medhekar, Bote, & Deshmukh (2013) tersebut mengklasifikasikan data menjadi 5 kategori. Akurasi dari metode ini sangat bergantung kepada algoritma dan data tu sendiri. Dalam penelitian yang akan dilakukan ini, metode LORENS yang memiliki algoritma yang cukup komples akan dibandingkan dengan metode NBC yang memiliki algoritma yang

lebih sederhana, serta metode *binary logistic regression* yang merupakan *base classifier* dari metode LORENS. Perbandingan ketiga metode tersebut diharapkan mampu memberikan hasil yang baik.

1.2 Rumusan Masalah

Metode klasifikasi untuk prediksi gen normal dan tidak normal dari data DNA microarray telah menjadi perhatian bagi pakar dibidang biologi molekular. Setelah diteliti oleh Ohwada (2015) menggunakan metode *Random Forest*, kasus ini akan diteliti dengan metode *Naive Bayes Classifier, Binary Logistic Regrssion* dan *Logistic Regression Ensembles* (LORENS). Dalam penelitian ini akan diteliti mengenai metode dan model klasifikasi yang terbaik dalam mengklasifikasikan gen normal dan gen yang terpengaruh penyakit Alzheimer. Pendekatan untuk prediksi klasifikasi pada kasus ini tidak menggunakan pendekatan inferensial. Pendekatan inferensial kurang terpercaya untuk menangani kasus dengan data yang besar, karena cenderung menghasilkan kesimpulan menolak hipotesa *null* pada tahap pengujian parameter. Oleh karena itu, pada kasus ini digunakan metode *Naive Bayes Classifier* dan *Logistic Regression Ensembles* (LORENS) yang tidak memerlukan pengujian parameter. Namun kelemahan kedua metode ini adalah tidak dapat memberikan model intepretatif yang dapat mengintepretasikan hubungan antara variabel prediktor dengan variabel respon. Perbandingan kedua metode diperlukan untuk memilih metode terbaik berdasarkan ketepatan klasifikasi yang diperoleh.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang dijelaskan diatas, berikut ini adalah tujuan penelitian dai penelitin ini.

1. Menghitung dan menganalisis hasil klasifikasi dan ketepatan klasifikasi gen yang terkait penyakit Alzheimer menggunakan metode *Naive Bayes Classifier*.

2. Menghitung dan menganalisis hasil klasifikasi dan ketepatan klasifikasi gen yang terkait sindrom Alzheimer menggunakan metode *Binary Logisic Regression*.
3. Menghitung hasil klasifikasi dan ketepatan klasifikasi gen yang terkait penyakit Alzheimer menggunakan metode *Logistic Regression Ensembles* (LORENS).
4. Memilih metode klasifikasi terbaik dari hasil analisis mengunakan metode *Naive Bayes Classifier*, *Binary Logistic Regression* dan *Logistic Regression Ensembles* (LORENS).

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah *pre-processing* dan *feature selection* data telah dilakukan pada penelitian sebelumnya.

(halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

2.1 *Naïve Bayes Classifier*

Naïve Bayes Classifier merupakan sebuah metode pengklasifikasi probabilitas sederhana yang menerapkan Teorema Bayes dengan asumsi ketidaktergantungan yang tinggi. Konsep dasar dari metode ini adalah teorema Bayes, dimana didalam statisik teorema ini banyak digunakan untuk menghitung peluang. Bila diketahui $A_1, A_2, ..., A_n$ adalah sebuah kejadian yang independen dalam ruang sampel Ω, sehingga $\sum_{i=1}^{n} A_i = \Omega$. B merupakan sebuah kejadian random dan kejadian $A_1 \cap B, A_2 \cap B, ..., A_n \cap B$ merupakan partisi dalam B (Gorunescu, 2011).

$$P(B) = \sum_{i=1}^{n} P(A_i \cap B) \tag{2.1}$$

Bila $P(A_i) > 0$ saat $i = 1, 2, ..., n$ maka $P(A_i \cap B) = P(B \mid A_i)P(A_i)$.

$$P(B) = \sum_{i=1}^{n} P(\mathrm{B} \mid A_1)P(A_i) \tag{2.2}$$

Kejadian random $A_1, A_2, ..., A_n$ dan B adalah partisi dari ruang sampel Ω. Jika $P(B) > 0$ dan $P(A_i) > 0$ untuk $i = 1, 2, ..., n$, maka:

$$P(A_i \mid B) = \frac{P(B \mid A_i)P(A_i)}{\sum_{i=1}^{n} P(B \mid \mathrm{A}_i)P(A_i)} \tag{2.3}$$

$P(A_i \mid B)$ merupakan *posterior probability* karena nilai $P(A_i \mid B)$ bergantung pada nilai B. $P(A_i)$ disebut *prior probability* karena nilainya tidak bergantung pada nilai B, sedangkan $P(B \mid A_i)$ adalah fungsi *likelihood* dan $P(B)$ merupakan keterangan.

Metode *Naïve Bayes Classifier* menggunakan konsep dari teorema Bayes. Bila diberikan $\{A_1, A_2, ..., A_n\}$ adalah atribut yang digunakan untuk menentukan kelas C, dengan menggunakan teorema Bayes maka perhitungan *posterior probability* untuk setiap kelas C adalah sebagai berikut (Gorunescu, 2011).

$$P(C_j \mid A_1, A_2, ..., A_n) = \frac{P(A_1, A_2, ..., A_n \mid C_j) P(C_j)}{\sum_{i=1}^{n} P(A_1, A_2, ..., A_n)} \qquad (2.4)$$

Apabila kelas tersebut memaksimalkan nilai $P(C_j \mid A_1, A_2, ..., A_n)$ atau memaksimalkan nilai $P(A_1, A_2, ..., A_n \mid C_j)$, maka kelas tersebut yang dipilih. Berdasarkan persamaan diatas, diperlukan perhitungan $P(A_1, A_2, ..., A_n \mid C_j)$. Setiap atribut diasumsikan independen untuk setiap kelas C. Apabila terdapat atribut yang memiliki sifat kuantitatif atau kontinyu, maka $P(A_i \mid C_j)$ dihitung dengan pendekatan distribusi normal.

$$P(A_i \mid C_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} \exp\left(-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \qquad (2.5)$$

Dugaan peluang $P(A_i \mid C_j)$ dapat dihitung untuk setiap atribut A_i dan kelas C_j, sehingga data baru dapat diklasifikasikan kedalam kelas C_k jika peluang yang didapat merupakan yang terbesar diantara peluang lainnya.

2.2 *Binary Logistic Regression*

Salah satu metode klasifikasi dasar adalah *Logistic Regression*. *Logistic Regression* dengan kasus menggunakan dua kelas respon bernama *Binary Logistic Regression*. *Binary Logistic Regression* merupakan suatu metode analisis data yang berguna untuk mencari sebuah hubungan variabel respon *y* yang bersifat biner dengan variabel prediktor *x* yang bersifat polikotomus (Hosmer dan Lemeshow, 2000). Variabel respon (y) dari regresi logistik biner terdiri dari 2 kategori yaitu "sukses" dan "gagal", dimana notasi dari $y = 1$ untuk kategori "sukses" dan $y = 0$ untuk kategori "gagal". Sehingga variabel respon y mengikuti distribusi Bernoulli untuk setiap observasi tunggalnya. Fungsi probabilitas untuk setiap observasinya adalah sebagai berikut:

$$f(y) = \pi^{y}(1-\pi)^{1-y}; y = 0,1 \tag{2.6}$$

dimana apabila $y = 0$ maka $f(y) = 1-\pi$ dan $y = 1$ maka $f(y) = \pi$, sehinga didapatkan fungsi regresi logistik sebagai berikut.

$$f(z) = \frac{e^{z}}{1+e^{z}}; \mathrm{z} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p \tag{2.7}$$

dimana p adalah banyak variabel prediktor. Nilai $f(z)$ terletak antara 0 dan 1 untuk setiap nilai z yang diberkan, karena nilai z sendiri terletak antara $-\infty$ dan ∞. Model regresi logistik tersebut sebenarnya menggambarkan sebuah probabilitas dari suatu objek. Model regresi logistiknya adalah sebagai berikut.

$$\pi(x) = \frac{e^{\beta_0 + \beta_1 x_1 + ... + \beta_p x_p}}{1+e^{\beta_0 + \beta_1 x_1 + ... + \beta_p x_p}} \tag{2.8}$$

Pendugaan parameter regresi dapat diuraikan dengan menggunakan transformasi logit dari persamaan $\{\pi(x)\}\{1+e^{(\beta_0+\beta_1 x_1+\cdots+\beta_p x_p)}\} = e^{(\beta_0+\beta_1 x_1+\cdots+\beta_p x_p)}$, sehingga didapatkan persamaan sebagai berikut.

$$g(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \qquad (2.9)$$

Pada regresi logistik, variabel respon dapat dituliskan sebagai $y = \pi(x) + \varepsilon$ dimana ε memiliki nilai $\varepsilon = 1 - \pi(x)$ dengan peluang $\pi(x)$ jika y = 1 atau $\varepsilon = -\pi(x)$ dengan peluang $1 - \pi(x)$ jika y = 0. Kedua kemungkinan tersebut mengikuti distribusi binomial dengan rata-rata nol dan varians $(\pi(x))(1-\pi(x))$.

Dalam regresi logistik, estimasi parameter dapat dilakukan dengan *Maximum Likelohood*. Estimasi parameter β diestimasi dengan memaksimumkan fungsi *likelihood* dan mensyaratkan data harus mengikuti distribusi tertentu, dalam regresi logistik adalah distribusi Bernoulli. Fungsi probabilitas untuk setiap pasangan bebas dan terikat pada setiap pengamatan adalah sebagai berikut.

$$f(x_i) = \pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i}; y_i = 0,1 \qquad (2.10)$$

dengan,

$$\pi(x_i) = \frac{e^{\left(\sum_{j=0}^{p}\beta_j x_j\right)}}{1+e^{\left(\sum_{j=0}^{p}\beta_j x_j\right)}} \qquad (2.11)$$

Dimana apabila nilai j = 0 maka nilai $x_{ij} = x_{i0} = 1$. Sehingga fungsi *likelihood* merupakan gabungan dari fungsi distribusi masing-masing data.

$$l(\beta) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} \pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i}$$

$$= \left\{\prod_{i=1}^{n}(1-\pi(x_i))\right\}\left\{\prod_{i=1}^{n} e^{(\log(\frac{\pi(x_i)}{1-\pi(x_i)})^{y_i})}\right\}$$

$$= \left\{ \prod_{i=1}^{n} (1 - \pi(x_i) \right\} e^{\left\{ \sum_{i=1}^{n} y_i \log(\frac{\pi(x_i)}{1-\pi(x_i)})^{y_i} \right\}}$$

$$= \left\{ \prod_{i=1}^{n} \frac{1}{1 + e^{\sum_{j=0}^{p} \beta_j x_{ij}}} \right\} e^{\left\{ \sum_{i=1}^{n} y_i \log(e^{\sum_{j=0}^{p} \beta_j x_{ij}}) \right\}} \qquad (2.12)$$

$$= \left\{ \prod_{i=1}^{n} (1 + e^{\sum_{j=0}^{p} \beta_j x_{ij}}) \right\} e^{\left\{ \sum_{j=0}^{p} (\sum_{i=1}^{1} y_i x_{ij}) \beta_j \right\}}$$

Agar lebih mudah, fungsi *likelihood* tersebut dimaksimumkan dalam bentuk log $l(\beta)$ dan dinyatakan dengan $\mathrm{L}(\beta)$.

$$L(\beta) = \log l(\beta) = \sum_{j=0}^{p} \left(\sum_{i=1}^{n} y_i x_{ij} \right) \beta_j - \sum_{i=1}^{n} \log \left(1 + e^{\sum_{j=0}^{p} \beta_j x_{ij}} \right) \qquad (2.13)$$

Nilai β maksimum diperoleh dari turunan $L(\beta)$ terhadap β dan hasilnya adalah sama dengan nol.

$$\frac{\partial L(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} y_i x_{ij} - \sum_{i=1}^{n} x_{ij} \left(\frac{e^{\sum_{j=0}^{p} \beta_j x_{ij}}}{1 + e^{\sum_{j=0}^{p} \beta_j x_{ij}}} \right)$$

$$= \sum_{i=1}^{n} y_i x_{ij} - \sum_{i=1}^{n} x_{ij} \hat{\pi}(x_i) = 0; \; j = 1, 2, \cdots p \qquad (2.14)$$

Selanjutnya setelah diperoleh parameter hasil estimasi adalah melakukan uji signifikansi terhadap koefisien β secara univariat terhadap variabel respon. Hipotesis pengujian parsial koefisien β adalah sebagai berikut.

$H_0 : \beta_i = 0$

$H_1 : \beta_i \neq 0; i = 1, 2, \cdots, p$

Dengan statistik uji sebagai berikut.

$$W = \frac{\hat{\beta}_i}{SE(\hat{\beta}_i)} \tag{2.15}$$

Statistik uji *W* atau *Wald* mengikuti distribusi normal, sehingga apabila nilai $|W| > Z_{\alpha/2}$ maka H_0 ditolak.

2.3 *Logistic Regression Classification By Ensembles From Random Partition* (LR CERP)

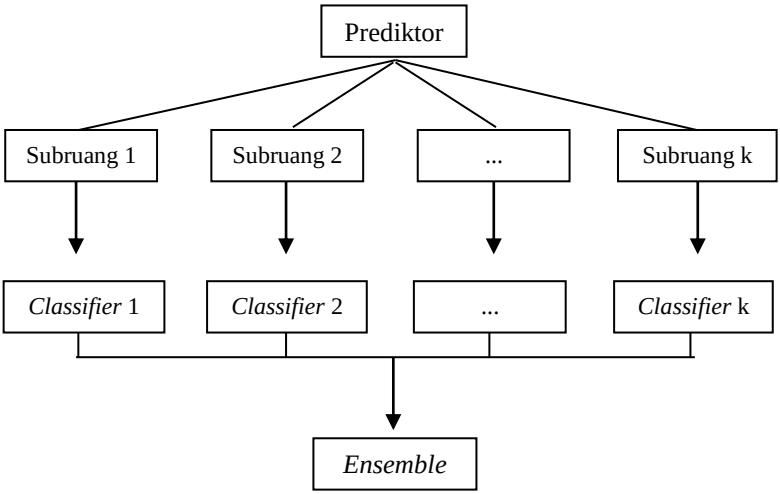
LR CERP (*Logistic Regression Classification By Ensembles From Random Partition*) adalah pasangan dari C-T CERP (*Classification Tree Classification By Ensembles From Random Partition*) yang menggunakan regresi logistik sebagai basis pengkasifikasi. Algoritma ini mempartisi ruang prediktor secara random menjadi sub-sub ruang yang saling *mutually exclusive* dengan ukuran yang sama. Misalnya Θ adalah sebuah ruang prediktor yang dipartisi menjadi *K* sub ruang $(\theta_1, \theta_2, ..., \theta_k)$ yang saling *mutually exclusive* dengan ukuran yang sama sehingga dapat diasumsikan tdak terdapat bias dalam pengambilan prediktor pada masing-masing sub ruang.

Berdasarkan *base classifier* regresi logistik diatas, performa CERP sangat tergantung oleh banyaknya variabel prediktor yang digunakan dalam satu partisi. Partisi yang optimal dapat diperoleh dari persamaan berikut ini.

$$K = \frac{6 \times p}{n} \tag{2.16}$$

dimana p adalah banyak variabel prediktor dan n adalah banyaknya observasi. Jika ukuran n lebih besar dari ukuran p, partisi yang optimal dapat didapatkan dengan membagi data sebanyak i menjadi $\frac{p}{i}$, dimana i adalah bilangan integer yang

kurang dari *n*. $K = \frac{p}{i}$ yang menghasilkan akurasi tertinggi merupakan jumlah partisi yang optimal. Berikut ini adalah bagan yang menggambarkan konsep *Logistic Regression Classification by Ensembles from Random Partition.*



Gambar 2.1 Bagan Konsep LR CERP

Model klasifikasi akan dibentuk pada tiap-tiap subruang dengan model regresi logistik, dimana model regresi logistik memiliki kelemahan pada pemilihan variabel. LR CERP berguna meningkatkan akurasi dengan cara mengombinasikan hasil klasifikasi pada masing-masing sub ruang yang terbentuk. Hal tersebut disebabkan jumlah prediktor dalam satu subruang lebih daripada banyaknya pengamatan. Kombinasi beberapa model regresi logistik yang dilakukan LR CERP berguna untuk meningkatkan akurasi dengan mengambil rata-rata nilai prediksi yang dihasilkan dalam satu *ensemble*. Nilai prediksi yang dihasilkan dari semua *base classifiers* dirata-rata dan dikategorikan menjadi 0 atau 1 berdasarkan *threshold* (Lim, 2007).

2.3 *Logistic Regression Ensemble*

LORENS dikembangkan oleh Lim, Ahn, Moon dan Chen pada tahun 2010 dengan menggunakan regresi logistik sebagai *base classifier* dan berdasarkan algoritma LR CERP. Dalam rangka meningkatkan akurasi prediksi, LORENS mengombinasikan hasil model regresi logistik untu mendapatkan satu *classifier* yang kuat dibanding metode agregasi kompleks lainnya. LORENS menggunakan prosedur yang sama dengan LR CERP, namun disini LORENS mengulangi prosedur LR CERP beberapa kali sampa terbentuk beberapa *ensemble*. LORENS mempartisi ruang prediktor Θ yang dipartisi menjadi *K* subruang $(\theta_1, \theta_2, ..., \theta_k)$ yang sama. Subruang dipilih secara acak berdasarkan distribusi yang sama, diasumsikan tidak terdapat bias pada saat pengambilan prediktor pada masing-masing subruang. Model regresi yang terbentuk pada masing-masing ruang dilakukan tanpa melalui seleksi variabel. Dengan melakukan pengacakan ini, diharapkan probabilitas yang sama pada pada masing-masing *classifier* pada satu *ensemble* dan juga error klasifikasi yang hampir sama.

Peningkatan akurasi dalam satu *ensemble* yang dihasilkan LORENS didapatkan dengan mengombinasikan nilai prediksi dari model-model regresi logistik pada masing masing partisi yang didapat. Dengan mengulangi prosedur LR CERP, LORENS mendapatkan kombinasi rata-rata ataupun nilai terbanyak yang menghasilkan akurasi yang hampir sama. Rata-rata menghasilkan nilai sedikit lebih unggul daripada nilai terbanyak, sehingga LORENS lebih baik menggunakan nilai rata-rata. Dengan menggunakan prosedur LR CERP, LORENS menghasilkan beberapa *ensemble* dengan patisi acak yang berbeda-beda pula. Dari beberapa *ensemble* yang terbentuk, diambil nilai terbanyak diantaranya. Berdasarkan nilai tersebut didapatkan satu akurasi umum. Nilai akurasi tersebut telah ditingkatka dengan sumbangsih dari beberapa *ensemble* yang dibangun.

Kelebihan LORENS selanjutnya adalah dalam hal penentuan *threshold*. Pada umumnya *Threshold* yang digunakan dalam

klasifikasi dengan respon biner adalah 0,5. Apabila proporsi kelas 0 dan 1 tidak seimbang, akurasi klasifikasi tidak akan baik. *Threshold* yang optimal dibutuhkan untuk menyeimbangkan *sensitifity* dan *spesificity*. Berikut merupakan rumus untuk menghitung *threshold* optimal dari LORENS.

$$Threshold = \frac{p + 0,5}{2} \tag{2.17}$$

p adalah probabilitas pengamatan yang berada di kelas positif. Berikut merupakan tahapan dalam proses klasifikasi.

1. Membentuk model logit dari data *training*.
2. Memasuka data *testing* ke dalam model logit, sehingga diperoleh nilai probabilitas.
3. Mengklasifikasikan pengamatan data *testing*. Jika nilai probabilitasnya lebih besar daripada nilai *threshold* maka pengamatan masuk ke dalam kelas positif, sebaliknya jika nilai probabilitasnya lebih kecil daripada nilai *threshold* maka pengamatan masuk ke dalam kelas negatif.
4. Membandingkan kelas aktual dengan prediksi klasifikasi.
5. Mengelompokkan hasil perbandingan ke dalam kelompok TP, TN, FP, dan FN.

TP (*True Positive*) adalah total ekspresi gen positif yang tepat terprediksi ke dalam kelas positif. TN (*True Negative*) adalah total ekspresi gen negatif yang tepat terprediksi ke dalam kelas negatif. FP (*False Positive*) adalah total ekspresi gen negatif yang terprediksi ke dalam kelas positif. FN (*False Negative*) adalah total ekspresi gen positif yang terprediksi ke dalam kelas negatif. Berikut merupakan tabel yang menunjukan prediksi klasifikasi dan kelas aktual.

Tabel 2.1 Tabel Tabulasi Silang Klasifikasi Aktual dan Klasifikasi Prediksi

		Kelas Aktual	
		p (+)	*n* (-)
Kelas Prediksi	*p* (+)	*True Positive*	*False Positive*
	n (-)	*False Negative*	*True Negative*

Untuk menghitung ketepatan prediksi klasifikasi, dapat dihitung dengan cara membagi jumlah prediksi yang tepat dengan total jumlah prediksi. Rumus untuk menghitung ukuran ketepatan klasifikasi adalah sebagai berikut (Catal, 2010).

$$Sensitivity = \frac{TP}{(TP + FN)} \tag{2.18}$$

$$Specificity = \frac{TN}{(FP + TN)} \tag{2.19}$$

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{2.20}$$

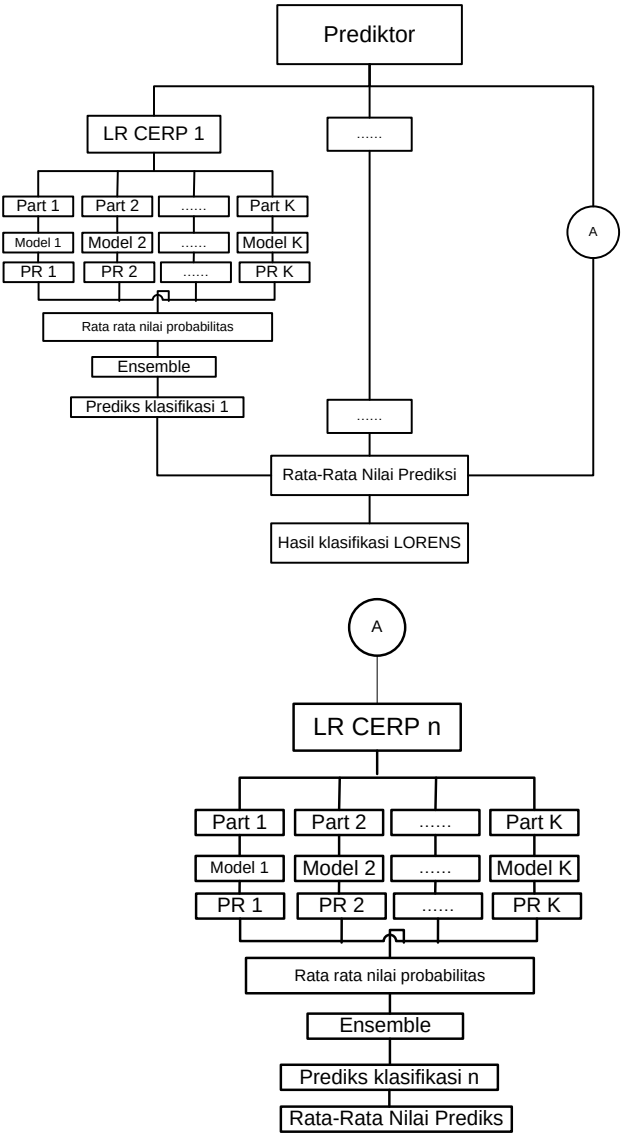
Sedangkan untuk kesalahan klasifikasi dapat dinyatakan dengan FPR (*False Positive Rate*), FNR (*False Negative Rate*), dan *Error*. Berikut adalah rumus untuk mendapatkan nilai kesalahan klasifikasi.

$$FPR = \frac{FP}{(\mathrm{T}\,P + FN)} \tag{2.21}$$

$$FNR = \frac{FN}{(TP + FN)} \tag{2.22}$$

$$Error = \frac{(FP + FN)}{(TP + FP + TN + FN)} \tag{2.23}$$

Gambar dibawah ini merupakan bagan yang menggambarkan konsep dari LORENS.



Gambar 2.2 Bagan Konsep LORENS

LORENS mempunyai kelebihan bebas dari asumsi dimensi data, karena LORENS melakukan partisi secara acak terhadap prediktornya. Dalam hal komputasi, LORENS lebih unggul daripada LR CERP yang masih menggunakan *tree algorithm* (algoritma pohon). Keakuratan metode dapat menjadi lebih baik dengan dua keunggulan LORENS tersebut diatas (Lee dkk., 2013).

2.4 C*ross Validation*

Terdapat beberapa metode untuk mengevaluasi performa sebuah model dalam melakukan prediksi melalui data *testing* dan data *training*, diantaranya adalah *Holdout* dan *Cross Validation* (Witten, Frank, & Hall, 2001). Metode *Holdout* menggunakan dua-pertiga dari data untuk digunakan menjadi data *training* dan menggunakan sisanya sebagai data *testing*. Ada kemungkinan sampel yang diambil tidak representatif, karena ada peluang setiap kelas dalam data tidak terwakili. Untuk memeriksa apakah sampel yang diambil representatif atau tidak, yaitu dengan cara menyeimbangkan proporsi masing-masing kelas untuk data *testing* dan data *training*. Apabila ada satu kelas yang tidak terwakili dalam data *training*, *classifier* tidak dapat terbentuk dengan baik untuk melakukan klasifikasi dalam data *testing*. Pengambilan sampel secara random harusnya memperhatikan dan menjamin bahwa sampel yang diambil sudah cukup mewakili masing-masing kelas yang ada. Salah satu cara agar setiap kelas dapat terwakili dalam data *trainig* dan data *testing* adalah dengan melakukan stratifikasi. Berikut merupakan langkah sederhana untuk melakukan stratifikasi.

1. Memisahkan data berdasarkan kelasnya.
2. Mengambil sampel dari masing-masing kelas dengan proporsi yang tepat.
3. Menggabungkan sampel dari masing-masing kelas yang terpilih.

Metode *cross validation* membagi data menjadi k *folds* atau partisi yang memiliki jumlah yang seimbang. Masing-masing partisi berperan sebagai data *traning* sekaligus data *testing* pada

saat gilirannya. Jelasnya, metode *cross validation* menggunakan satu partisi data sebagai data *testing* dan $k-1$ sisanya sebagai data *training*. Prosedur ini terus berulang sampai semua partisi data telah menjadi data *testing*. Metode atau prosedur ini dikenal dengan nama *k fold cross validation*. Namun apabila prosedur stratifikasi juga dilakukan, metode ini disebut *stratified k fold cross validation*. Misalnya digunakan 10 *folds* untuk metode *cross validation*. Pertama, data dibagi secara acak menjadi 10 bagian dengan proporsi sama. Selanjutnya metode *cross validation* ini dijalankan sebanyak 10 kali dengan data *training* yang berbeda. Dimana setiap set data memiliki jumlah yang sama dengan set data yang lainnya. Pengujian telah dilakukan dengan menggunakan data yang berbeda dan teknik belajar yang berbeda pula, kesimpulanya 10 *folds* merupakan *folds* terbaik untuk mendapatkan kesalahan yang terbaik. Metode 10 *folds cross validation* telah menjadi metode standar dalam *machine learning* dan *data mining*. Metode evaluasi ini juga menunjukan dengan penggunaan stratifikasi dapat meningkatkan akurasi prediksi. Berikut merupakan langkah dalam melakukan metode *Cross Validation*.

1. Memisahkan variabel respon berdasarkan kelasnya.
2. Membagi keseluruhan pengamatan menjadi 10 partisi pada masing-masing kelas.
3. Menggabungkan kedua kelas pada bagian yang sama.
4. Mengunakan salah satu partisi sebagai data *testing* dan menggunakan bagian kedua sampai ke sepuluh menjadi data *training* pada *folds* yang pertama. Terus berlanjut sampai *folds* ke sepuluh menjadi data *testing*.

Prosedur ini menggunakan sebanyak 100 kali algoritma pada dataset, dengan tujuan mendapatkan performa yang baik. Pembagian data menjadi $\frac{9}{10}$ data *training* dan $\frac{1}{10}$ bagian lainnya sebagai data *testing*, dapat juga diadopsi pada metode *Holdout* (Witten dkk., 2011).

2.5 AUC (*Area Under Curve*)

Salah satu ukuran dasar yang digunakan mengukur dan mengevaluasi performa klasifikasi adalah sensitivitas dan spesifitas. Satu model klasifikasi biner memiliki sepasang sensitivitas dan spesifitas. Apabila dalam suatu kasus klasifikasi digunakan beberapa model klasifikasi, akan timbul masalah dalam hal pemilihan model dan metode terbaik. Hal tersebut dikarenakan terdapat beberapa pasang sensitifitas dan spesifitas dari model klasifikasi yang digunakan. Masalah tersebut dapat diatasi dengan menggunakan kurva ROC (*Receiving Operating Characteristic*). Kurva ROC merupakan representasi dari hubungan antara sensitifitas dan spesifitas secara grafis (Erke & Pattynama, 1998).

Kurva ROC sering digunakan untuk mengevaluasi metode klasifikasi karena mempunyai kemampuan menyeluruh dan cukup baik (Chou dkk., 2010). Pada kurva ROC, sensitivitas (*true positive rate*) diplot dalam fungsi 1-spesifitas (*false positive rate*) untuk poin *cut off* yang berbeda-beda. Setiap titik pada kurva ROC merupakan pasangan dari sensitivitas dan spesifitas yang sesuai dengan batasan keputusan tertentu. Sebuah tes dengan diskriminasi sempurna memiliki plot yang melewati sudut kiri atas dari kurva ROC (sensitivitas 100% dan spesifitas 100%. Semakin dekat plot ROC ke sudut kiri atas, maka semakintinggi pula akurasi dari keseluruhan tes (Zweig & Campbell, 1993).

Metode yang umum digunakan untuk menghitung performasi klasifikasi adalah dengan menghitung luas daerah dibawah kurva ROC. Area dibawah kurva ROC biasa disebut *Area Under The ROC Curve* (AUC). Nilai AUC berada diantara 0 dan 1. Apabila nilai AUC semakin mendekati 1, maka model klasifikasi yang terbentuk semakin akurat. Kurva ROC yang baik berada disebelah atas dari garis diagonal (0,0) dan (1,1), sehingga tidak ada nilai AUC yang lebih kecil dari 0,5.

Perhitungan AUC dilakukan melalui beberapa pendekatan, yang paling banyak digunakan adalah *trapezoidal method*. Pendekatan tersebut berbasis metode geometris yang berdasarkan

interpolasi kinier antara masing-masing titik pada kurva ROC. Khusus untuk kasus biner, nilai AUC dapat didekati dengan nilai *Balanced Accuracy* (Bekkar, Djemaa, & Alitouch, 2013).

$$AUC = \frac{1}{2}(sensitivity + specificity) \qquad (2.24)$$

Kategori berdasarkan nilai AUC dapat disajikan ke dalam tabel berikut.

Tabel 2.2 Kategori Pengklasifikasian Model Berdasarkan Nilai AUC

Nilai AUC	Model Diklasifikasikan Sebagai
0,91-1,00	*Excelent* (Sempurna)
0,81-0,90	*Very Good* (Sangat baik)
0,71-0,80	*Good* (Baik)
0,61-0,70	*Fair* (Cukup)
0,51-0,60	*Poor* (Lemah)

Sumber : Bekkar dkk. (2013)

2.6 DNA Microarray Alzheimer

Perubahan atau mutasi gen dalam DNA tertentu dapat menjadi indikator untuk terjadinya penyakit tertentu. Namun sangat sulit untuk mengembangkan tes untuk mendeteksi mutasi ini. Pada kasus kanker payudara herideter dan kanker ovarium misalnya, mutasi pada gen BRCA1 dan BRCA2 menyebabkan menyebabkan 60% dari jumlah kasus tersebut. Peneliti menyimpulkan bahwa tidak hanya satu mutasi saja yang meyebabkan kasus tersebut, namun ditemukan lebih dari 800 mutasi yang berbeda pada gen BRCA1 saja. Microarray DNA merupakan alat yang digunakan untuk menentukan apakah DNA dari suatu makhluk hidup tertentu mengandung mutasi gen seperti pada gen BRCA1 dan BRCA2. Microarray DNA berupa chip yang terdiri dari lempengan kaca kecil yang terbungkus plastik. Setiap chip berisi ratusan bahkan ribuan fragmen DNA (Anonim, 2015).

Aplikasi microarray DNA banyak digunakan dalam deteksi kanker, dimana sel kanker mengalami abnormalitas dalam mengekspresikan gennya. Teknologi ini memungkinkan peneliti

untuk dapat melihat tahapan perkembangan sel kanker dengan melihat level ekspresinya terhadap probe spesifik yang terdapat pada chip microarray. Analisis ini menggunakan sampel DNA normal dan DNA yang abnormal (kanker, tumor, alzheimer dan penyakit degeneratif lainnya). Kedua sampel tersebut diamplifikasi dan diberi pewarna fluorescent yang berbeda pada masing-masing sampel DNA. Pada umumnya, DNA normal akan diberi warna hijau dan DNA yang abnormal akan diberi warna merah. Setelah itu akan dilakukan proses hibridisasi, pada proses ini DNA akan memancarkan cahaya sesuai zat pewarna yang telah diberikan. Apabila DNA membawa ekspresi normal dan abnormal, maka akan muncul warna lain, bisasanya warna kuning. Namun apabila tidak ada DNA yang mampu melakukan hibridisasi dengan probe, zat pewarna tidak akan terekspresi dan biasanya terlihat hitam. Warna-warna tersebut yang kemudian dibaca oleh detektor dan kemudian diubah menjadi data grafik sehingga dapat diolah secara kuantitatif (Cowell & Hawthorn, 2007).

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

Data yang digunakan dalam penelitian ini adalah data sekunder yag berasal dari penelitian yang dilakukan oleh Nishiwaki dkk (2015). Dalam penelitian tersebut telah dilakukan seleksi variabel menggunakan metode *Random Forest*. Dari hasil penelitian tersebut didapatkan 20 variabel dari 11.555 variabel dengan *importance scores* tertinggi. Sehingga dalam penelitian ini hanya menggunakan 20 variabel terbaik yang didapatkan dari penelitian sebelumnya. Terdapat dua jenis ekspresi gen dalam data penelitian ini, yaitu gen normal dan gen abnormal (Alzheimer).

3.2 Variabel Penelitian

Variabel yang digunakan dalam penelitian ini terdiri dari 1 variabel respon biner (Y) yaitu Gen Normal dan Gen AD (*Alzheimer Disease*) dan 20 variabel prediktor. Berikut merupakan struktur data yang akan digunakan dalam penelitian ini.

Tabel 3.1 Struktur Data Penelitian

	Ekspresi Gen	Gene Symbol			
No	Normal/AD	WWOX	TAGLN3	⋯	MT1H
1	Normal	$X_{1,1}$	$X_{2,1}$	⋯	$X_{20,1}$
2	AD	$X_{1,2}$	$X_{2,2}$	⋯	$X_{20,2}$
3	Normal	$X_{1,3}$	$X_{2,3}$	⋯	$X_{20,3}$
⋮	⋮	⋮	⋮	⋱	⋮
178	AD	$X_{1,178}$	$X_{2,178}$	⋯	$X_{20,178}$

Variabel penelitian yang digunakan adalah 20 kode protein gen. Kode-kode protein gen tersebut mempunyai fungsinya masing-masing dalam menyelidiki penyakit tertentu, dalam hal ini

penyakit Alzheimer. Tabel dibawah ini merupakan kode-kode protein gen beserta nama ilmiahnya.

Tabel 3.2 Variabel Penelitian

Variabel	*Gene title*	*Symbol*
X1	WW domain containing oxidoreductase	WWOX
X2	transgelin 3	TAGLN3
X3	collagen, type V, alpha 2	COL5A2
X4	metallothionein 1F	MT1F
X5	Ets2 repressor factor	ERF
X6	apelin receptor	APLNR
X7	WNT inhibitory factor 1	WIF1
X8	glial fibrillary acidic protein	GFAP
X9	inositol-trisphosphate 3-kinase B	ITPKB
X10	collectin sub-family member 12	COLEC12
X11	lactate dehydrogenase A	LDHA
X12	solute carrier family 16, member 5	SLC16A5
X13	neuritin 1	NRN1
X14	synaptotagmin V	SYT5
X15	versican	VCAN
X16	neuronal pentraxin II	NPTX2
X17	hippocalcin	HPCA
X18	RAB6A, member RAS oncogene family	RAB6A
X19	WW domain containing transcription	WWTR1
X20	metallothionein 1H	MT1H

3.3 Langkah Analisis

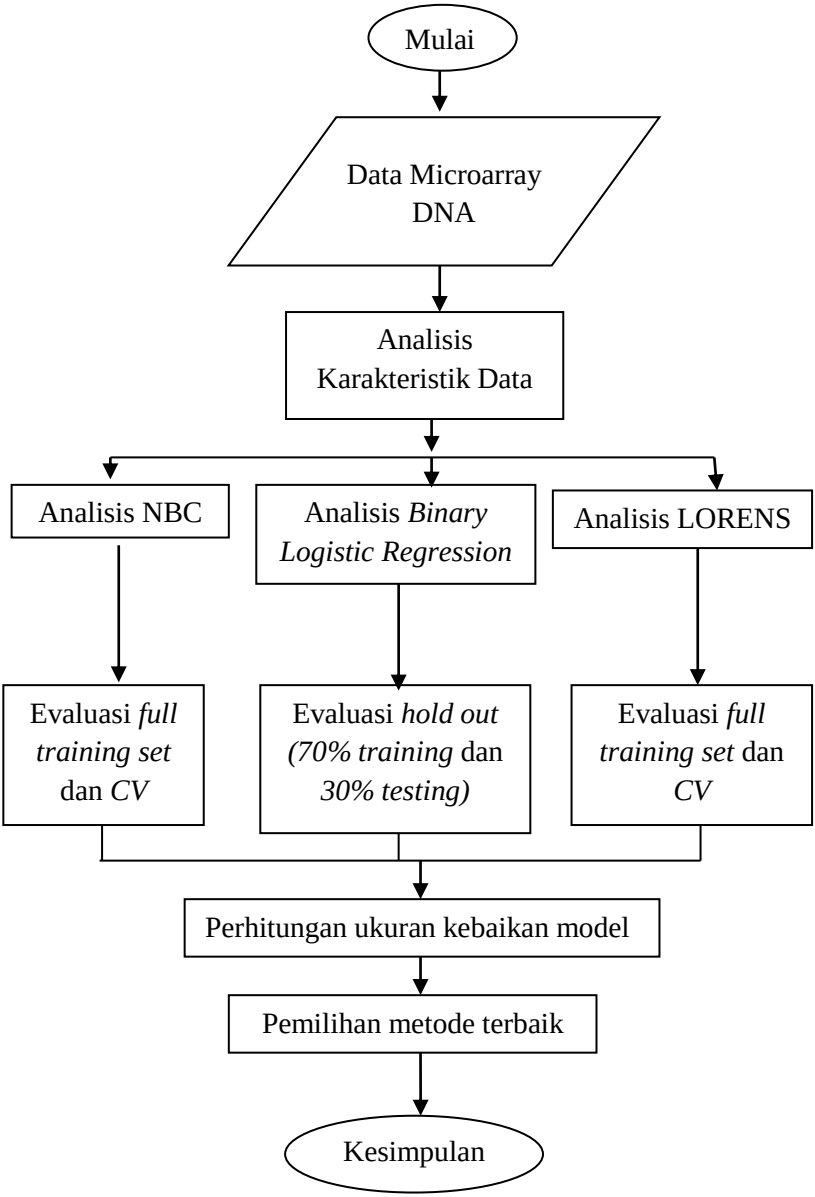
Dalam penelitian ini analisis klasifikasi akan dilakukan dengan metode *Naïve Bayes Classifier* dan LORENS. Kedua metode klasifikasi tersebut akan dievaluasi menggunakan metode *Holdout* dan *Cross Validation*. Langkah-langkah analisis dalam penelitian ini adalah sebagai berikut:

1. Membuat analisa deskriptif terhadap data.
2. Melakukan analisis klasifikasi menggunakan metode *Naïve Bayes Classifier* dengan prosedur evaluasi *full training set.*
3. Melakukan analisis klasifikasi menggunakan metode *Naïve Bayes Classifier* dengan prosedur evaluasi *Cross Validation.*
   a. Melakukan stratifikasi pada data.
   b. Membagi data menjadi 10 bagian yang sama.
   c. Mengambil sampel satu bagian data sebagai data *testing* dan menggunakan 9 bagian data lainnya sebagai data *training*.
   d. Menghitung rata-rata dan standar deviasi dari setiap prediktor pada masing-masing kelas pada data *training*.
   e. Menghitung peluang tiap masing-masing variabel prediktor pada masing-masing kategori.
   f. Menghitung *posterior probability* pada data testing.
   g. Menentukan kelas prediksi pada data testing.
4. Melakukan analisis klasifikasi menggunakan metode *Binary Logistic Regression* dengan prosedur evaluasi *full training set.*
5. Melakukan analisis klasifikasi menggunakan metode *Binary Logistic Regression* dengan prosedur evaluasi *Cross Validation..*
6. Melakukan analisis klasifikasi menggunakan metode LORENS dengan prosedur evaluasi *full training set.*
7. Melakukan analisis klasifikasi menggunakan metode *Logistic Regression* dengan prosedur evaluasi *Stratified 10-folds Cross Validation.*
   a. Melakukan stratifikasi pada data.
   b. Membagi data menjadi 10 bagian yang sama.
   c. Mengambil sampel satu bagian data sebagai data *testing* dan menggunakan 9 bagian data lainnya sebagai data *training*.
   d. Menentukan banyak partisi (k) serta banyak *ensemble* (n) dimana $k = 1, 2, ..., 5$ dan $n = 10$.

e. Mempartisi variabel prediktor menjadi (k) subruang partisi dari data *training.*
f. Menyusun model LR masing-masing subruang partisi dari data *training*.
g. Mendapatkan nilai akurasi prediksi dari masing-masing model untuk semua pengamatan dari data *testing.*
h. Menghitung nilai rata-rata dari semua nilai prediksi untuk masing-masing pengamatan.
i. Mengulangi langkah a hingga e sampaiterbentuk *n ensemble*.
j. Mencari nilai prediksi terbanyak masing-masing pengamatan diantara semua *ensemble*.
k. Menghitung nilai *threshold* optimal.
l. Membandingkan hasil dari langkah g dengan nilai *threshold* 0,5 dan *threshold* optimal.
m. Mengulangi semua langkah hingga semua data telah diperlakukan sebagai data *training* dan data *testing.*

8. Menghitung nilai *accuracy, sensitivity*, *specificity* dan AUC dari semua model yang terbentuk.
9. Memilih metode terbaik dari hasil analisis pada langkah 2, 3, 4, 5 dan 6 berdasarkan ketepatan klasifikasi terbaik.
10. Membuat kesimpulan dari hasil analisis yang telah dilakukan.

Berdasarkan langkah analisis diatas, diagram alir (*flow chart*) analisis dalam penelitian ini secara umum dapat diilustrasikan seperti pada gambar berikut ini.



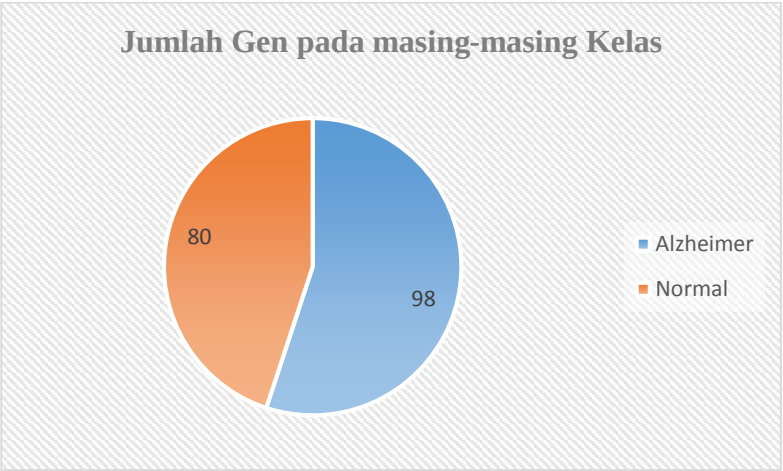
Gambar 3.1 Diagram Alir Penelitian

(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS DAN PEMBAHASAN

4.1 Analisis Karakteristik Data

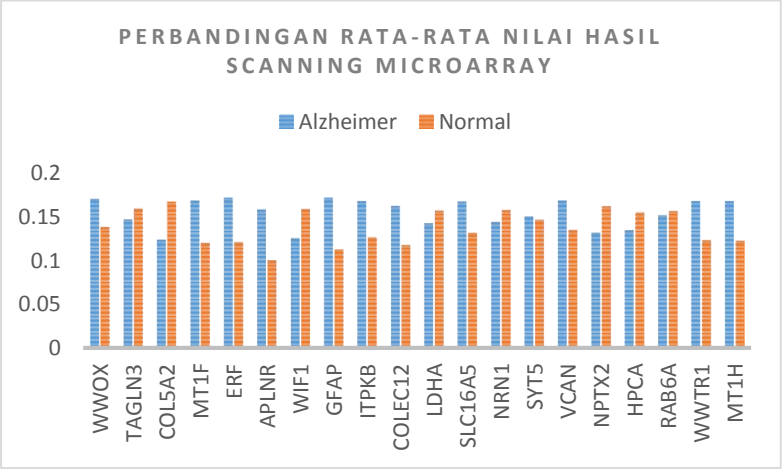
Penelitian mengenai klasifikasi gen ini ditujukan untuk mengetahui deteksi dini penyakit Alzheimer dengan memanfaatkan data yang diperoleh dari *microarray* DNA. Penelitian ini menggunakan variabel respon yang bersifat biner, yaitu gen yang bersifat normal dan gen yang terindikasi penyakit Alzheimer. Sedangkan variabel prediktor yang digunakan adalah 20 gen yang diduga sebagai *identifier* penyakit Alzheimer. Sebanyak 178 sampel ekspresi gen dari hasil *scanning microarray* DNA digunakan untuk memprediksi klasifikasi gen normal dan Alzheimer. Gambar 4.1 berikut menunjukan proporsi kelas respon yang digunakan dalam penelitian ini.



Gambar 4.1 Perbandingan Jumlah Gen Antara Gen Normal dan Alzheimer

Gambar 4.1 menunjukan bahwa sebanyak 98 gen Alzheimer atau sebanyak 55% gen Alzheimer digunakan sebagai sampel dalam penelitian ini, sedangkan gen normal yang digunakan dalam penelitian ini adalah 80 atau 45% dari total sampel gen yang digunakan. Hal tersebut menunjukan bahwa kelas respon

yang digunakan dalam penelitian ini tidak *balance*, sehingga diperlukan perhitungan *threshold* optimal dalam proses klasifikasi yang akan dilakukan. Klasifikasi dilakukan dengan menggunakan 20 variabel prediktor yang telah diseleksi dari 11.555 variabel. Variabel prediktor yang digunakan merupakan komponen penyusun DNA. Nilai variabel prediktor yang digunakan dalam penelitian ini adalah nilai hasil *scanning microarray*. Gambar dibawah ini merupakan grafik perbandingan nilai rata-rata nilai hasil *scanning microarray* ke-20 variabel prediktor antara gen normal dan gen yang terkait dengan sindrom Alzheimer.



Gambar 4.2 Perbandingan Rata-Rata Nilai *Scanning Microarray* Gen Normal dan Alzheimer

WWOX (WW *Domain Containing Oxidoreductase*) merupakan salah satu kode protein dari sebuah gen. Kode protein gen ini mengkodekan anggota keluarga protein dehydrogenase rantai pendek. Kode protein gen ini berfungsi sebagai supresor gen tumor dan penyakit degeneratif lainnya, ekspresi yang dihasilkan dari *scanning microarray* DNA kode protein gen ini mampu menunjukan beberapa penyakit. Pada kode protein gen WWOX menunjukan gen Alzheimer memiliki nilai yang lebih tinggi daripada gen normal. Hal tersebut sesuai dengan fungsi

kode protein gen WWOX yang mampu membedakan gen yang normal dan gen yang cacat, dalam hal ini terkait dengan sindrom Alzheimer. Variabel TAGLN3 (*Transgelin* 3) pada umumnya diketahui sebagai penekan tumor. Rata-rata nilai hasil *scanning microarray* pada kode protein TAGLN3 lebih tinggi gen normal daripada gen Alzheimer. Bersesuaian dengan fungsi TAGLN3 sebagai penekan tumor, gen penyakit Alzheimer pun menunjukan nilai yang lebih rendah daripada gen yang normal. COL5A2 (*Collagen Type V Alpha 2 Chain*) merupakan kode gen untuk kelimpahan kolagen fibriliar. Pada variabel ini, gen normal memiliki nilai hasil *scanning microarray* untuk kode gen yang lebih tinggi daripada gen Alzheimer. ERF (ETS2 *Repressor Factor*) merupakan salah satu kode protein gen, dimana penyakit yang terkait dengan ERF meliputi *Craniosynostosis* 4 dan *Craniosynostosis.* Aktifitas yang terkait dengan protein gen ini mencakup faktor transkripsi, aktifitas pengikatan spesifik DNA dan mentranskrip spesimen. Gen yang terkait sindrom Alzheimer memiliki nilai hasil *scanning microarray* untuk kode protein gen yang lebih tinggi daripada gen normal. Sedangkan pada variabel APLNR (*Apelin Receptor*), GFAP (*Glial Fibrillari Acidic Protein*), ITPKB (*Inositol-triphosphate 3-kinase B*), COLEC12 (*Collectin sub-family member 12*), SLC16A5(*Solute Carrier Family 16*), SYT5 (*Synaototagmin V*), VCAN (*Versican*), WWTR1 (*WW domain containing transcription regulator 1*), MTIH (*Metallotjionein*), nilai hasil *scanning microarray* gen yang terkait Alzheimer memiliki nilai yang lebih tingi daripada gen normal. Berbeda dengan variabel WIF1 (WNT *Inhibitory Factor* 1), LDHA (*Lactate Dehydrogenase* A), NRN1 (*Neuritin* 1), NPTX2 (*Neuronal Pentraxin* II), HPCA (*Hippocalcin*), RAB6A (RAS *Oncogene family*), dimana rata-rata nilai hasil *scanning microarray* pada gen yang terkait Alzheimer justru lebih rendah daripada gen normal.

4.2 Pengujian Proporsi Variabel Respon

Pengujian proporsi antar kelas dalam variabel respon diperlukan untuk mengetahui apakah proporsi antara gen Alzheimer dan gen normal seimbang atau tidak. Pengujian proporsi ini mengunakan uji Z statistik, dengan hipotesis sebagai berikut.

H_0 : Proporsi Kelas Alzheimer sama dengan 0,5

H_1 : Proporsi Kelas Alzheimer tidak sama dengan 0,5

Pengambilan keputusan dilakukan dengan mengunakan *p-value*. Berikut ini merupakan tabel perbandingan *p-value* dengan alpha 0,05.

Tabel 4.1. Perbandingan *P-value* dan alpha

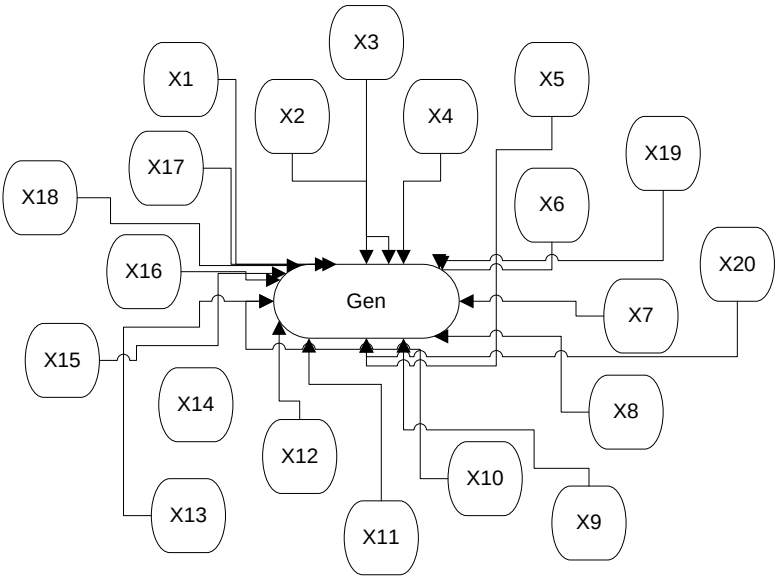
Z	***P-value***	**Alpha**
1,349	0,202	0,05

Karena *p-value* lebih besar daripada alpha 0,05, maka keputusannya adalah gagal menolak H_0. Artinya proporsi antara kelas gen Alzhiemer dan Normal dikatakan sama secara statistik. Hal tersebut mengakibatkan nilai *accuracy* dan AUC yang digunakan akan memiliki nilai yang hampir sama. Namun proporsi antara kelas Alzheimer dan normal belum tentu sama pada tiap *fold Cross Validation*, sehingga AUC tetap digunakan dalam penelitian ini.

4.3 Analisis *Naive Bayes Classifier Full Training Set*

Analisis dengan menggunakan *Naïve Bayes Classifier* merupakan pendekatan untuk menyelesaikan masalah klasifikasi yang sederhana. Klasifikasi dilakukan dengan memanfaatkan peluang yang didapat dari perhitungan menggunakan variabel prediktor. Analisis ini menggunakan keseluruhan data untuk membentuk model dan mengunakan keseluruhan variabel prediktor untuk menjadi data *testing*. Hal tersebut bertujuan untuk menguji seberapa baik model yang terbentuk. Data *testing* akan diklasifikasikan kedalam kelas respon dengan nilai peluang

terbesar. Apabila nilai peluang kelas respon 1 lebih besar daripada kelas respon 0, maka data tersebut masuk kedalam kelas 1, begitu juga sebaliknya. Dengan menggunakan 20 prediktor, gambar struktur *Naïve Bayes* memprediksi gen dapat diilustrasikan kedalam gambar berikut.



Gambar 4.3 Struktur *Naïve Bayes* Klasifikasi Gen

Selanjutnya adalah melakukan perhitungan nilai probabilitas pada masing-masing pasangan data. nilai probabilitas yang terbesar memiliki kecendrungan lebih besar terhadap prediksi klasifikasi variabel respon. *Naïve Bayes* mengharuskan variabel prediktornya diskrit, apabila ada beberapa atau semua variabel prediktor yang bersifat kontinyu maka $P(A_i \mid C_j)$ harus dihitung dengan pendekatan distribusi normal. Sebelum menghitung nilai $P(A_i \mid C_j)$, diperlukan perhitungan rata-rata dan standar deviasi setiap variabel prediktor pada masing-masing kelas respon. Tabel

dibawah ini merupakan perhitungan rata-rata dan standar deviasi pada semua variabel prediktor dan kelas respon.

Tabel 4.2 Rata-Rata dan Standar Deviasi Setiap Prediktor dan Kelas

Gen	Rata-Rata		Standar Deviasi	
	C=0	C=1	C=0	C=1
WWOX	0,1381	0,1707	0,0464	0,0679
TAGLN3	0,1592	0,1470	0,0765	0,0633
COL5A2	0,1672	0,1239	0,0929	0,0769
MT1F	0,1200	0,1686	0,0519	0,0933
ERF	0,1210	0,1720	0,0555	0,0840
APLNR	0,1005	0,1585	0,0602	0,1218
WIF1	0,1587	0,1259	0,0955	0,0855
GFAP	0,1129	0,1719	0,0475	0,0965
ITPKB	0,1266	0,1679	0,0753	0,0721
COLEC12	0,1176	0,1625	0,0552	0,1046
LDHA	0,1570	0,1426	0,0745	0,0782
SLC16A5	0,1317	0,1676	0,0657	0,0728
NRN1	0,1576	0,1439	0,0749	0,0743
SYT5	0,1465	0,1504	0,0848	0,0713
VCAN	0,1350	0,1687	0,0626	0,0674
NPTX2	0,1620	0,1319	0,0997	0,0647
HPCA	0,1548	0,1348	0,0966	0,0761
RAB6A	0,1567	0,1518	0,0678	0,0650
WWTR1	0,1231	0,1678	0,0699	0,0811
MT1H	0,1229	0,1680	0,0463	0,0937

Nilai rata-rata dan standar deviasi pada Tabel 4.2 diatas berguna untuk menghitung nilai peluang tiap kategori pada data *testing* yang digunakan. Analisis ini menggunakan keseluruhan data *training* untuk digunakan sebagai data *testing*. Perhitungan peluang dilakukan untuk setiap variabel predktor yang digunakan. Terdapat 20 nilai peluang untuk masing-masing kelas. Nilai peluang tersebut selanjutnya berguna untuk menghitung peluang pasangan data *testing* untuk menentukan kelas data *testing* tersebut. Perhitungan nilai peluang tiap kategori pada data *testing* pertama dilakukan seperti pada Tabel 4.3 dibawah ini.

Tabel 4.3 Peluang Tiap Kategori Pada Data *Testing* Pertama

Variabel Prediktor	$P(X_1 \mid C = 0)$	$P(X_1 \mid C = 1)$
WWOX	0,3698	0,1638
TAGLN3	0,9975	0,9281
COL5A2	0,0878	0,0050
MT1F	0,4011	0,1759
ERF	0,0004	$5{,}6.10^{-5}$
APLNR	$2{,}5.10^{-8}$	$3{,}1.10^{-6}$
WIF1	0,4107	0,1456
GFAP	0,0001	0,0018
ITPKB	0,0010	$8{,}3.10^{-8}$
COLEC12	0,9226	0,8656
LDHA	0,4089	0,3177
SLC16A5	0,0004	$3{,}6.10^{-6}$
NRN1	0,1841	0,1159
SYT5	0,0089	0,0016
VCAN	0,0657	0,0038
NPTX2	0,9428	0,4809
HPCA	0,5952	0,2605
RAB6A	0,1847	0,1323
WWTR1	0,0202	0,0005
MT1H	0,6539	0,3689

Tabel 4.3 diatas menunjukan nilai peluang untuk masing-masing variabel prediktor pada data *testing* pertama. Untuk peluang parsial pada masing-masing prediktor pada tabel diatas dapat diketahui bahwa nilai peluang variabel prediktor X_1 yaitu WWOX memiliki nilai peluang terbesar pada kelas respon berkategori 0, dimana kategori tersebut merupakan gen normal. Hat tersebut berarti variabel WWOX pada data *testing* pertama memiliki peluang masuk kedalam kelas gen normal daripada kelas gen Alzheimer. Agar dapat mengklasifikasikan data *testing* menggunakan seluruh variabel prediktor, diperlukan perhitungan *posterior probability* untuk setiap pasangan data *testing.* Persamaan dibawah ini merupakan ilustrasi perhitungan *posterior probability* pada data *testing* pertama.

$$P(X_1, X_2, ..., X_{20} \mid C_0) = P(x_{1,1} \mid C_0).P(x_{1,2} \mid C_0)...P(x_{1,20} \mid C_0)$$

$$= 0,3697 \text{x} 0,9975 \text{x} \cdots \text{x} 0,6539$$

$$= 1,183.10^{-30}$$

$$P(X_1, X_2, ..., X_{20} \mid C_1) = P(x_{1,1} \mid C_1).P(x_{1,2} \mid C_1)...P(x_{1,20} \mid C_1)$$

$$= 0,3697 \text{x} 0,9975 \text{x} \cdots \text{x} 0,6539$$

$$= 9,22.10^{-43}$$

Berdasarkan ilustrasi perhitungan *posterior probability* pada data testing pertama, didapatkan hasil bahwa data *testing* pertama diklasifikasikan kedalam kelas 0, yaitu gen normal. Hal tersebut dilakukan karena nilai peluang dari kelas gen normal lebih tinggi daripada kelas gen Alzheimer. Perhitungan yang sama dilakukan sampai dengan data *testing* terakhir.

Tabel 4.4 Perhitungan *Posterior Probability* pada Data *Testing*

No	Kelas 0 (Gen Normal)	Kelas 1 (Gen Alzheimer)	Kelas Prediksi
1	$1,18.10^{-30}$	$9,22.10^{-43}$	0
2	$3,43.10^{-30}$	$4,73.10^{-37}$	0
3	$2,44.10^{-21}$	$5,21.10^{-25}$	0
4	0,4873	$1,09.10^{-10}$	0
5	$1,49.10^{-5}$	$1,03.10^{-9}$	0
6	$3,86.10^{-15}$	$1,11.10^{-26}$	0
7	$4,49.10^{-7}$	$4,14.10^{-8}$	0
8	$3,78.10^{-12}$	$1,4.10^{-11}$	1
9	$8,08.10^{-7}$	0,0003	1
10	$6,81.10^{-23}$	$5,7410^{-16}$	1
⋮	⋮	⋮	⋮
175	0,0032	$1,59.10^{-5}$	0
176	0,0035	$8,68.10^{-6}$	0
177	0,0033	$9,22.10^{-6}$	0
178	0,0022	$7,04.10^{-6}$	0

Berdasarkan Tabel 4.4, didapatkan nilai *posterior probability* untuk keseluruhan data *testing*. Dari nilai tersebut selanjutnya dapat dilakukan prediksi klasifikasi. Kelas prediksi didapatkan dari nilai peluang terbesar pada masing-masing kelas. Apabila kelas 0 ata gen normal memiliki nilai peluang yang lebi tinggi maka data *testing* tersebut masuk kedalam kelas 0 atau gen normal, begitu juga sebaliknya. Setelah semua data *testing* berhasil diprediksi, selanjutnya dilakukan pengelompokan *True Positive, True Negative, False Positif* dan *False Negative*. Hasil pengelompokan tersebut dapat dilihat pada tabel dibawah ini.

Tabel 4.5 Tabulasi Silang pada Analisis *Naïve Bayes Classifier*

		Kelas Aktual	
		+	-
Kelas Prediksi	+	59	39
	-	12	68

Bedasarkan Tabel 4.15, dapat diketahui bahwa analisis menggunakan *Naïve Bayes Classifier* dan menggunakan 20 variabel memberikan hasil 59 ekspresi gen Alzheimer yang tepat diprediksi sebagai gen Alzheimer dan 68 ekspresi gen normal yang diprediksi sebagai gen normal. Sedangkan terdapat 39 ekspresi gen Alzheimer yang diprediksi sebagai gen normal dan 12 ekpresi gen normal yang diprediksi sebagai gen Alzheimer. Ukuran *accuracy* dari hasil prediksi diatas adalah 0,71348 dengan *sensitivity* sebesar 0,83098 dan *specificity* 0,6355. Analisis menggunakan *Naïve Bayes Classifier* telah mampu memprediksi ekpresi gen dengan ketepatan sebesar 71,35%. Metode ini cukup baik mengkasifikasikan gen Alzheimer tepat kedalam kelasnya sebesar 83,1%, sedangkan untuk memprediksi gen normal tepat terklasifikasi kedalam gen normal sebesar 63,55%.

4.4 Analisis *Naive Bayes Classifier Cross Validation*

Setelah dilakukan analisis *Naïve Bayes full training set*, selanjutnya dilakukan analisis *Naïve Bayes* dengan evaluasi *Cross Validation*. Analisis ini bertujuan untuk mengkonfirmasi apakah model yang terbentuk dari metode ini baik digunakan untuk data

testing. Analisis ini dilakukan dengan membagi data menjadi 10-*folds* yang seimbang. Salah satu *fold* akan dijadikan data *testing* dan Sembilan sisanya akan menjadi data *training*. Hal tersebut berulang sampai semua *fold* pernah menjadi data *testing*. Berikut ini merupakan tabulasi silang dari analisis *Naïve Bayes Classifier*.

Tabel 4.6 Tabulasi Silang Kelas Aktual dan Prediksi *Naïve Bayes Classifier Cross Validation*

		Kelas Aktual	
		+	-
Kelas Prediksi	+	40	65
	-	15	58

Berdasarkan Tabel 4.6, dapat diketahui bahwa terdapat 40 gen Alzheimer yang terprediksi tepat kedalam kelas Alzheimer. 58 gen normal yang tepat terprediksi kedala gen normal, sedangkan kesalahan dari prediksi sangat besar. Kesalahan dalam memprediksi gen Alzheimer namun masuk kedalam kelas normal terdapat 65 gen dan kesalahan dalam memprediksi gen normal namun masuk kedalam kelas Alzheimer terdapat 15 gen. berikut ini merupakan perhitungan ketepatan klasifikasi.

Tabel 4.7. Ukuran Kebaikan Klasifikasi *Naïve Bayes Classifier Crosss Validation*

Accuracy	*Sensitivity*	*Specifity*	*AUC*
0.55056	0.72727	0.47154	0.59941

Ukuran *accuracy* dari analisis diatas termasuk rendah bila dibandingkan dengan akurasi dari metode lainnya, hanya sekitar 55,06% dengan *sensitivity* sebesar 72,73% dan *specifity* sebesar 47,15%. Analisis ini memiliki nilai AUC sebesar 0,5994, yang mengindikasikan bahwa analisis ini lemah dalam menangani kasus klasifikasi gen.

4.5 Analisis *Binary Logistic Regression Full Training Set*

Analisis menggunakan metode ini berguna untuk menggambarkan hubungan antara ekspresi gen dengan 20 variabel prediktor, yaitu kode pretein dari gen dalam DNA.

Evaluasi dalam analisis ini mmenggunakan prosedur *Full Training Set.* Dimana keseluruhan data akan digunakan sebagai data *training*, dan keseluruhan data pula yang digunakan sebagai data *testing* untuk menguji seberapa baik model yang terbentuk.

Koefisien parameter model regresi diestimasi menggunakan *Maximum Likelihood Estimation* dari data *training* yang digunakan untuk membangun model. Berikut ini adalah hasil estimasi parameter model *Binary Logistic* pada data *training*.

Tabel 4.8 Koefisien Parameter Awal Model *Binary Logistic Regression Full Training Set*

Parameter	Koefisien	*Wald*	*P-Value*
Intercept	-4,28	10,02	0,002
X_1	-1,3	0,01	0,905
X_2	-16,81	3,33	0,068
X_3	-14,95	8,77	0,003
X_4	17,16	4,56	0,033
X_5	10,9	3,44	0,064
X_6	-7,75	0,88	0,348
X_7	-17,22	6,89	0,009
X_8	21,5	5,44	0,02
X_9	5,7	0,42	0,515
X_{10}	49.82	6,41	0,011
X_{11}	-2,12	0,43	0,835
X_{12}	7,91	1,39	0,239
X_{13}	-4,23	0,3	0,586
X_{14}	0,14	0,00	0,983
X_{15}	-36,69	6,24	0,013
X_{16}	-14,4	4,881	0,027
X_{17}	-4,26	0,838	0,36
X_{18}	49,84	7,89	0,005
X_{19}	8,68	0,75	0,385
X_{20}	13,23	2,38	0,123

Dari Tabel 4.8 diatas, dengan menggunakan tingkat kesalahan 5% dan taraf keyakinan 95% pada proses eliminasi variabel, diketahui bahwa terdapat banyak variabel yang tidak signifikan. Oleh karena itu, diperlukan eliminasi satu per satu

variabel yang tidak signifikan sampai didapatkan koefisien model regresi logistik yang seluruhnya signifikan. Berikut ini merupakan tabel koefisien regresi logistik biner yang telah melalui eliminasi *backward*.

Tabel 4.9 Koefisien Parameter Terbaik Model *Binary Logistic Regression*

Parameter	Koefisien	*Wald*	*P-Value*
Intercept	-4,26	16,1	0,000
X_4	-10,26	8,93	0,003
X_5	15,39	7,26	0,007
X_6	9,94	8,5	0.004
X_8	-8,83	6,76	0,009
X_9	19,84	9,45	0,002
X_{11}	15,74	7,53	0,006
X_{16}	-18,29	6,05	0,014
X_{19}	11,25	7,09	0,008

Dari Tabel 4.9 diatas, dapat diketahui bahwa terdapat 12 variabel prediktor yang dieliminasi dari total 20 variabel prediktor. Seluruh koefisien parameter mempunyai *p-value* diatas batas kesalahan yang ditetapkan untuk mengeliminasi, yaitu 5%. Setelah koefisien parameter didapatkan, selanjutnya dapat dilanjutkan dengan menuliskan model yang terbentuk. Model regresi logistik biner yang terbentuk adalah sebagai berikut.

$$\pi(x) = \frac{e^{-4,26-10,26x_4+15,39x_5+9,94x_6+\cdots+11,25x_{19}}}{1+e^{-4,26-10,26x_4+15,39x_5+9,94x_6+\cdots+11,25x_{19}}} \qquad (4.1)$$

Meskipun lebih banyak variabel yang dieliminasi daripada yang digunakan, analisis tetap dilanjutkan guna menjawab tujuan penelitian untuk mendapatkan hasil prediksi, variabel prediktor pada data *testing* disubtitusikan kedalam model, sehingga didapatkan nilai probabilitasnya. Apabila nilai probabilitasnya lebih dari 0,5 maka gen tersebut diklasifikasikan kedala kelas Alzheimer, sebaliknya bila kurang dari 0,5 maka diklasifikasikan kedalam kelas normal. Berikut ini merupakan tabulasi silang hasil klasifikasi.

Tabel 4.10 Tabulasi Silang pada *Binary Logistic Regression*

		Kelas Aktual	
		+	-
Kelas Prediksi	+	68	6
	-	30	74

Berdasarkan Tabel 4.10, dapat diketahui bahwa dari 54 data *testing* yang digunakan, 26 gen Alzheimer tepat terprediksi kedalam kelas Alzheimer, 7 gen normal tepat terprediksi kedalam kelas gen normal dan sisanya tidak terklasifikasi dengan tepat. Dari tabulasi diatas, selanjutnya dapat dhitung nilai keteatan klasifikasi. Berikut ini merupakan perhitungan ketepatan klasifikasi.

Tabel 4.11 Ukuran Kebaikan Model *Binary Logistic Regression*

Model	*Accuracy*	*Sensitivity*	*Specificity*	AUC
Regresi Logistik	0,798	0,694	0,925	0,809

Model *Binary Logistic Regression* mampu memperoleh akurasi sebesar 79,8%. Dengan nilai *sensitivity* dan *specificity* sebesar 69,4% dan 92,5%. AUC yang dihasilkan oleh analisis ini adalah 80,9%. Model yang terbentuk dapat mengklasifikasikan variabel respon dengan baik.

4.6 Analisis *Binary Logistic Regression Cross Validation*

Salah satu prosedur evaluasi dalam klasifikasi adalah C*ross Validation.* Prosedur ini dapat mengevaluasi model yang terbentuk dengan baik, karena semua data diperlakukan sebagai data *training* dan data *testing*. Pada penelitian ini, *Cross Validation* dilakukan dengan menggunakan 10 *folds*. Pada masing-masing *fold* akan terbentuk satu model regresi logistik biner, sehingga dalam analisis ini akan terbentuk 10 model regresi logistik biner yang berbeda.

Proses analisis pada setiap *fold* sama dengan tahapan proses pada analisis *Binary Logistic Regression Full Training Set*. Pertama adalah estimasi koefisien parameter model, uji

signifikansi parameter sampai dengan didapatkan model terbaik dari masing-masing *fold*. Berikut ini adalah estimasi koefisien parameter model *Binary Logistic* pada *fold* pertama.

Tabel 4.12 Koefisien Parameter Awal Model *fold* ke-1 *CV Binary Logistic Regression*

Parameter	Koefisien	*Wald*	*P-Value*
Intercept	-4,77	7,01	0,008
X_1	1,7	0,01	0,905
X_2	-14,45	1,47	0,226
X_3	-16,46	6,3	0,012
X_4	13,92	2,51	0,113
X_5	7,41	1,13	0,287
X_6	-19,72	2,94	0,086
X_7	-18,16	4,33	0,038
X_8	38,97	7,26	0,007
X_9	9,42	0,81	0,368
X_{10}	21,08	5,77	0,016
X_{11}	-9,62	0,34	0,558
X_{12}	11,23	2,09	0,148
X_{13}	-7,61	0,617	0,432
X_{14}	-3,19	0,15	0,694
X_{15}	-42,32	5,14	0,023
X_{16}	-22,25	3,64	0,056
X_{17}	-8,46	1,61	0,205
X_{18}	70,71	8,19	0,004
X_{19}	11,18	0,86	0,355
X_{20}	15,93	2,43	0,119

Dari Tabel 4.12 diatas, dapat diketahui bahwa banyak variabel prediktor tidak memiliki pengaruh yang signifikan terhadap model. Langkah selanjutnya adalah melakukan eliminasi variabel prediktor. Eliminasi dilakukan dengan menggunakan prosedur *backward wald* dengan tingkat kesalahan 5%. Dengan demikian, variabel prediktor yang memiliki *p-value* terbesar akan dieliminasi, proses tersebut berulang sampai dengan seluruh variabel prediktor yang digunakan menghasilkan *p-value* kurang dari 0,05. Tabel 4.13 dibawah ini merupakan tabel koefisien

parameter dari model regresi logistik yang telah seluruhnya signifikan.

Tabel 4.13 Koefisien Parameter Terbaik Model *Fold* ke-1 *CV Binary Logistic Regression*

Parameter	Koefisien	*Wald*	*P-Value*
Intercept	-5,06	15,01	0,000
X_3	-12,42	9,90	0,002
X_8	25,84	9,47	0,002
X_{10}	16,64	5,80	0,016
X_{12}	-9,33	6,17	0,013
X_{15}	-15,67	4,05	0,044
X_{16}	-19,77	7,04	0,008
X_{18}	20,02	6,73	0,009
X_{20}	18,45	6,02	0,013

Berdasarkan Tabel 4.13 diatas, hanya terdapat 8 variabel prediktor yang digunakan dalam model regresi logistik. Variabel prediktor tersebut adalah variabel yang digunakan dalam model. Proses yang sama dilakukan untuk *fold* ke-2 sampai dengan *fold* ke-10. Setelah dilakukan proses yang sama sampai dengan *fold* ke-10, didapatkan model regresi logistik untuk masing-masing *fold*. Model yang terbentuk pada masing-masing *fold* merupakan model yang terbaik, dimana variabel yang digunakan adalah variabel prediktor yang telah terseleksi dengan menggunakan eliminasi *backward wald* dengan tingkat kesalahan 5%. Model *Binary Logistic Regression* pada masing-masing *fold* berguna untuk memprediksi klasifikasi pada data *testing* yang diberikan. Hasil prediksi klasifikasi pada setiap *fold* akan digabungkan menjadi satu kemudian dihitung ukuran ketepatan klasifikasi untuk *Binary Logistic Regression* dengan prosedur evaluasi 10 *fold Cross Validation*. Berikut ini merupkan model yang terbentuk dari masing-masing *fold*.

Tabel 4.14 Model *Binary Logistic Regression* Pada Seluruh *Fold*

Fold	Model *Binary Logistic Regression*
1	$\frac{e^{-5,06-12,42x_3+25,84x_8+16,64x_{10}+9,33x_{12}-15,67x_{15}-19,77x_{16}+20,02x_{18}+18,45x_{20}}}{1+e^{-5,06-12,42x_3+25,84x_8+16,64x_{10}+9,33x_{12}-15,67x_{15}-19,77x_{16}+20,02x_{18}+18,45x_{20}}}$
2	$\frac{e^{-4,39-9,32x_3+13,92x_4+9,49x_5-10,88x_7+21,06x_8+16,44x_{10}-17,42x_{15}+12,3x_{18}}}{1+e^{-4,39-9,32x_3+13,92x_4+9,49x_5-10,88x_7+21,06x_8+16,44x_{10}-17,42x_{15}+12,3x_{18}}}$
3	$\frac{e^{-1,81-16,19x_2+12,68x_3+16,92x_5-11,97x_7-14,94x_{16}+28,89x_{18}+20,7x_{20}}}{1+e^{-1,81-16,19x_2+12,68x_3+16,92x_5-11,97x_7-14,94x_{16}+28,89x_{18}+20,7x_{20}}}$
⋮	⋮
10	$\frac{e^{-3,33-24,66x_2-26,28x_3+23,72x_4+\cdots+38,44x_{19}}}{1+e^{-3,33-24,66x_2-26,28x_3+23,72x_4+\cdots+38,44x_{19}}}$

Dari model yang terbentuk pada masing-masing *fold*, selanjunya dapat dihitung *True Positive*, *True Negative*, *False Positive* dan *False Negative* dari seluruh *fold*. Dari perhitungan tersebut, dapat dilanjutkan dengan menghitung ukuran ketepatan klasifikasi model *Binary Logistic Regression* dengan prosedur evaluasi *Cross Validation*. Tabel dibawah ini merupakan tabel yang menunjukan ukuran ketepatan klasifikasi pada analisis *Binary Logistic Regression* dengan prosedur evaluasi *Cross Validation*.

Tabel 4.15 Tabulasi Silang pada *Binary Logistic Regression*

		Kelas Aktual	
		+	-
Kelas Prediksi	+	59	16
	-	46	57

Berdasarkan tabel 4.15, dapat diketahui bahwa terdapat 59 gen Alzheimer yang tepat terklasifikasi pada kelasnya dan 57 gen normal yang tepat terklasifikasi pada kelas gen normal. Sedangkan 46 gen Alzheimer tidak terprediksi secara tepat pada analisis ini, hal tersebut mengindikasikan bahwa kesalahan

klasifikasi pada analisis ini cukup besar. Perhitungan ukuran kebaikan model diperlukan untuk mengevaluasi model yang terbentuk. Berikut ini merupakan tabel yang menunjukan ukuran kebaikan model *Binary Logistic Regression* dengan evaluasi *Cross Validation*.

Tabel 4.16 Ukuran Kebaikan Model *Binary Logistic Regression CV*

Model	*Acc*	*Sens*	*Spec*	AUC
10 *Fold CV Binary Logistic*	0,652	0,562	0,781	0,671

Analisis *Binary Logistic Regression* dengan prosedur evaluasi *Cross Validation* mampu menghasilkan *accuracy* sebesar 65,2% dengan *sensitifity* 56,2% dan *specificity* sebesar 78,1%. Nilai AUC yang dihasilkan pada analisis ini sebesar 0,671, artinya model yang terbentuk belum cukup baik untuk menangani kasus klasifikasi gen yang terkait sindrom Alzheimer.

4.7 Analisis LORENS *Full Training Set*

Analisis *Logistic Regression Ensembles* (LORENS) merupakan pendekatan komputasional untuk menyelesaikan masalah klasifikasi. LORENS tidak memiliki asumsi apapun untuk dipenuhi. Demi mendapatkan hasil klasifikasi terbaik, analisis LORENS dilakukan beberapa kali dengan jumlah partisi yang berbeda pula. Dalam kasus ini, partisi yang dibentuk adalah sebanyak 2, 3, 4 dan 5 partisi. Kelebihan LORENS yang lain adalah dapat menemukan nilai *threshold* optimal, namun pada penelitian ini analisis LORENS dengan *threshold* 0,5 tetap akan digunakan sebagai perbandingan. Nilai *ensemble* yang akan digunakan dalam penelitian ini adalah sebesar 10, karena dari hasil penelitian sebelumnya dapat menghasilkan akurasi klasifikasi yang baik. Dengan ukuran *ensemble* 10, model yang didapatkan untuk 2 partisi adalah 20 model, 3 partisi adalah 30 model, 4 partisi adalah 40 model dan untuk 5 partisi adalah 50 model.

Variabel prediktor pada analisis LORENS dialokasikan ke dalam ruang-ruang partisi yang terbentuk. Pengalokasian sampel dilakukan dengan cara *random sampling*. Proses tersebut berulang

terus sampai dengan jumlah *ensemble* yang ditentukan. Dalam penelitian ini, 20 variabel prediktor yang digunakan akan dialokasikan ke dalam ruang-ruang partisi yang terbentuk. Berikut ini adalah analisis LORENS untuk masing-masing ruang partisi. Pada analisis LORENS *full training set*, didapatkan hasil paling baik menggunakan 5 partisi dengan *threshold* 0,5.

Analisis LORENS menggunakan 4 partisi dengan *threshold* 0,5 dan proses tersebut berulang sebanyak *ensemble* yang ditentukan, yaitu 10 kali membentuk 40 model regresi logistik. Tahap pertama dalam analisis LORENS adalah membagi variabel kedalam beberapa ruang partisi, partisi yang menghasilkan hasil terbaik pada penelitian ini adalah sebesar 4 partisi. Berikut adalah tabel pengalokasian variabel prediktor kedalam ruang partisi.

Tabel 4.17 *Random Sampling* Variabel Prediktor pada 4 Ruang Partisi *Threshold* 0,5

Variabel	Ensemble									
	1	2	3	4	5	6	7	8	9	10
WWOX	1	3	1	2	2	2	2	3	4	2
TAGLN3	4	3	3	4	3	4	3	4	4	1
COL5A2	4	1	2	4	2	1	3	3	2	2
MT1F	1	2	1	3	1	2	4	3	4	4
ERF	2	1	4	1	3	4	4	3	2	3
APLNR	1	4	4	1	4	1	3	4	1	1
WIF1	2	3	3	3	3	1	3	3	2	1
GFAP	1	1	3	4	4	3	2	2	3	4
ITPKB	3	3	4	4	3	4	3	4	1	3
COLEC12	4	4	2	3	1	3	1	1	4	1
LDHA	1	1	1	3	4	4	1	1	3	2
SLC16A5	4	3	4	2	4	3	1	4	1	3
NRN1	2	2	3	2	2	3	2	2	3	2
SYT5	3	2	1	1	2	2	4	1	1	1
VCAN	2	4	2	3	2	4	2	4	2	2
NPTX2	3	4	2	1	3	2	2	1	3	4
HPCA	3	1	1	2	4	2	4	1	1	3
RAB6A	3	2	2	1	1	1	1	2	3	4
WWTR1	2	4	4	2	1	3	4	2	2	3
MT1H	4	2	3	4	1	1	1	2	4	4

Tabel 4.17 menunjukan pengalokasian variabel prediktor ke dalam 4 ruang partisi. Pada *ensemble* pertama, variabel X_1, X_4, X_6, X_8 dan X_{11} merupakan variabel prediktor pada ruang partisi pertama. Variabel X_5, X_7, X_{13}, X_{15} dan X_{19} merupakan variabel prediktor pada ruang partisi kedua. Variabel X_9, X_{14}, X_{16}, X_{17} dan X_{18} merupakan variabel prediktor pada ruang partisi ketiga. Variabel X_2, X_3, X_{10}, X_{12} dan X_{20} merupakan variabel prediktor pada ruang partisi keempat. Cara pengalokasian variabel prediktor yang sama juga dilakukan untuk *ensemble* ke-2 hingga ke-10. Tiap *ensemble* akan terbentuk 4 model regresi logistik yang berbeda dengan variabel prediktor yang berbeda pula.

Tabel 4.18 Koefisien Model Regresi Logistik 4 Partisi *Threshold* 0,5

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-2,1	-2,2	-1,5	-2,3	-2,2	-1,5	-2,6	-0,4	-2,0	-1,2
Partisi ke-2	-1,8	-1,5	-0,6	-1,6	-1,2	-1,2	-1,8	-1,9	-1,5	-0,9
Partisi ke-3	-1,0	-1,2	-1,3	-1,7	-1,1	-2,6	-1,1	-1,9	-1,4	-2,2
Partisi ke-4	-2,4	-1,1	-2,8	-1,5	-2,4	-2,0	-2,5	-2,0	-2,1	-1,9
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
WWOX	5,1	9,1	6,7	5,3	9,0	7,8	10,6	5,2	4,3	11,3
TAGLN3	-2,2	-3,9	0,1	0,7	0,8	-3,9	2,8	-2,1	-4,8	-0,4
COL5A2	-10,6	-9,1	-8,4	-8,8	-9,7	-8,1	-6,1	-8,1	-8,6	-8,6
MT1F	4,6	6,9	10,0	12,7	5,6	10,6	9,0	12,1	5,5	4,4
ERF	10,4	10,5	9,7	12,7	8,6	10,8	10,2	9,2	11,6	10,0
APLNR	3,5	3,6	8,0	7,7	4,5	7,0	7,2	7,5	7,8	7,7
WIF1	-4,7	-5,1	-5,5	-6,8	-4,3	-5,5	-3,1	-3,2	-2,6	-5,6
GFAP	7,9	13,6	10,1	9,1	11,1	12,6	13,2	9,8	14,3	9,1
ITPKB	10,3	4,4	-0,4	3,2	6,9	1,6	9,2	4,1	4,0	-1,0
COLEC12	6,9	3,4	4,9	9,7	6,8	5,2	6,7	8,4	5,8	6,5
LDHA	-3,6	3,9	-2,9	0,3	-0,7	1,7	-4,8	0,8	-6,3	-2,0
SLC16A5	9,2	4,5	3,1	5,6	7,6	6,0	6,0	7,0	6,9	3,8
NRN1	-0,4	-7,7	-2,1	-2,5	-2,0	-3,2	-1,4	-7,8	-4,1	0,3
SYT5	2,8	2,4	2,0	-2,7	4,0	1,8	0,4	2,8	0,5	3,3
VCAN	1,1	3,6	9,8	-1,4	7,8	5,3	-2,8	0,1	2,2	6,1
NPTX2	-7,1	-5,0	-7,7	-4,6	-3,0	-6,5	-5,2	-4,9	-6,0	-10,2
HPCA	-3,8	-1,1	-3,6	-2,5	-3,2	-3,4	-3,8	-2,6	-2,6	-3,4
RAB6A	6,1	2,5	7,0	5,6	-3,8	4,3	0,5	4,6	13,9	4,8
WWTR1	7,8	4,7	2,2	6,8	2,5	0,7	4,0	2,7	9,9	7,7
MT1H	16,3	8,4	8,4	8,8	6,8	15,8	12,2	6,3	6,5	7.4

Catatan:

Partisi ke-1 dilambangkan warna	
Partisi ke-2 dilambangkan warna	
Partisi ke-3 dilambangkan warna	
Partisi ke-4 dilambangkan warna	

Tabel 4.18 menunjukan koefisien model regresi yang terbentuk pada kelima ruang partisi yang terbentuk dan masing-

masing *ensemble*. Pada masing-masing *ensemble* terbentuk 4 model dengan koefisien yang ada pada tabel 4.2 diatas. Koefisien variabel prediktor tiap model regresi logistik sesuai dengan pengalokasian variabel prediktor pada tabel 4.1. Berikut ini merupakan ilustrasi beberapa model regresi logistik yang terbentuk pada *ensemble* pertama.

$$\pi_1(x) = \frac{e^{-2,1+5,1x_1+4,6x_4+3,5x_6+7,9x_8-3,6x_{11}}}{1+e^{-2,1+5,1x_1+4,6x_4+3,5x_6+7,9x_8-3,6x_{11}}} \quad (1)$$

$$\pi_2(x) = \frac{e^{-1,8+10,4x_5-4,7x_7-0,4x_{13}+1,1x_{15}+7,8x_{19}}}{1+e^{-1,8+10,4x_5-4,7x_7-0,4x_{13}+1,1x_{15}+7,8x_{19}}} \quad (2)$$

$$\pi_3(x) = \frac{e^{-1,0+10,3x_9+2,8x_{14}-7,1x_{16}-3,8x_{17}+6,1x_{18}}}{1+e^{-1,0+10,3x_9+2,8x_{14}-7,1x_{16}-3,8x_{17}+6,1x_{18}}} \quad (3)$$

$$\pi_4(x) = \frac{e^{-2,4-2,2x_2-10,6x_3+6,9x_{10}+9,2x_{12}+16,3x_{20}}}{1+e^{-2,4-2,2x_2-10,6x_3+6,9x_{10}+9,2x_{12}+16,3x_{20}}} \quad (4)$$

Persamaan (1) diatas merupakan model regresi logistik yang terbentuk dari variabel prediktor pada ruang partisi pertama, persamaan (2) terbentuk dari variabel prediktor pada ruang partisi kedua sampai dengan persamaan (4) dibangun dari variabel prediktor pada ruang partisi keempat pada *ensemble* pertama. Cara membaca yang sama dilakukan pada *ensemble* ke-2 sampai *ensemble* ke-10. Model regresi logistik yang terbentuk pada masing-masing ruang partisi dan *ensemble* berguna untuk menghitung nilai probabilitas setiap pasangan data *testing* guna memprediksi kelas data *testing* itu berada. Nilai probabilitas diperoleh dengan mensubtitusikan data *testing* kedalam model regresi logistik yang terbentuk. Nilai probabilitas akhir diperoleh dari rata-rata nilai probabilitas pada tiap ruang partisi masing-masing *ensemble*. Apabila nilai probabilitas akhir dari data *testing* lebih besar dari *threshold* yang ditentukan, maka data *testing* tersebut masuk kedalam kelas 1 yang merupakan gen Alzheimer. Namun apabila nilai probabilitas akhirnya lebih kecil daripada

threshold yang ditentukan maka data *testing* tersebut masuk kedalam kelas 0 yang merupakan gen normal. Prediksi data *testing* dapat dilakukan setelah model didapatkan, dimana prediksi yang dihasilkan menggunakan rata-rata dari nilai probabilitas dari masing-masing ruang partisi dan seluruh *ensemble*. Tabel dibawah ini merupakan nilai rata-rata probabilitas dan keputusan kelas klasifikasi pada model yang terbentuk di 4 ruang partisi dengan menggunakan *threshold* 0,5.

Tabel 4.19 Rata-Rata Nilai Probabilitas Pada 4 Partisi *Threshold* 0,5

No	Nilai Rata-Rata Probabilitas	Keputusan
1	0,19	0
2	0,23	0
3	0,38	0
4	0,46	0
5	0,40	0
6	0,22	0
7	0,49	1
8	0,47	0
9	0,59	1
10	0,71	1
⋮	⋮	⋮
174	0,45	0
175	0,46	0
176	0,46	0
177	0,46	0
178	0,46	0

Berdasarkan tabel 4.19 diatas, dapat diketahui pada pasangan data *testing* pertama menghasilkan rata-rata nilai probabilitas sebesar 0,19, karena nilai tersebut kurang dari *threshold* yang ditentukan maka pasangan data *testing* pertama masuk kedalam kelas 0, yaitu gen normal. Pasangan data *testing* kedua menghasilkan nilai rata-rata probabilitas sebesar 0,23, maka keputusannya adalah masuk kedalam kelas 0. Proses tersebut berulang sampai dengan pasangan data *testing* terakhir, yaitu pasangan data *testing* ke-178. Nilai rata-rata probabilitasnya

sebesar 0,46, ketusan yang dihasilkan adalah data *testing* tersebut masuk kedalam kelas 0. Setelah semua data *testing* diprediksi, selanjutnya adalah menghitung TP, TN, FP dan FN. Hasil klasifikasi pada data testing dengan menggunakan LORENS 4 ruang partisi dan *threshold* 0,5 dapat ditampilkan dalam tabulasi silang seperti berikut.

Tabel 4.20 Tabulasi Silang pada LORENS 4 Ruang Partisi *Threshold* 0,5

		Kelas Aktual	
		+	-
Kelas Prediksi	+	67	11
	-	31	69

Berdasarkan tabel 4.20, dapat diketahui bahwa terdapat 67 ekspresi gen Alzheimer yang tepat diprediksi sebagai gen Alzheimer (*True* Positive) dan 69 ekspresi gen normal yang diprediksi sebagai gen normal (*True Negative*). Sedangkan terdapat 31 ekspresi gen Alzheimer yang diprediksi sebagai gen normal (*False Positive*) dan 11 ekpresi gen normal yang diprediksi sebagai gen Alzheimer (*False Negative*).

Tabel 4.21 Ukuran Kebaikan Model LORENS *Full Training Set*

Model LORENS	*Accuracy*	*Sensitivity*	*Specificity*	AUC
2 Part Thres 0,5	0,753	0,865	0,673	0,769
2 Part Thres Opt	0,736	0,870	0,651	0,760
3 Part Thres 0,5	0,747	0,863	0,667	0,765
3 Part Thres Opt	0,742	0,871*	0,657	0,764
4 Part Thres 0,5	0,764*	0,859	0,690*	0,774*
4 Part Thres Opt	0,742	0,871*	0,657	0,764
5 Part Thres 0,5	0,758	0,848	0,687	0,767
5 Part Thres Opt	0,725	0,845	0,645	0.745

Catatan : * adalah nilai terbesar

Berdasarkan tabel 4.21 diatas, dapat diketahui pada analisis LORENS dengan *full training set* menghasilkan nilai akurasi terbaik pada saat 4 partisi *threshold* 0,5, yaitu 0,764 atau 76,4%.

Berbeda dengan nilai *accuracy*, nilai *sensitivity* terbaik diperoleh pada saat membagi data menjadi 3 partisi dan 4 partisi dengan *threshold* optimal, yaitu sebesar 0,871. Selaras dengan nilai *accuracy*, nilai *specificity* terbaik diperoleh pada saat membagi data menjadi 4 partisi dengan *threshold* 0,5, yaitu sebesar 0,69. Pemilihan model terbaik menjadi sulit apabila kelas respon yang digunakan tidak *balance*, yang menyebabkan nilai *specificity* dan *sensitivity* tidak selaras. Untuk menangani hal tersebut diperlukan satu nilai yang dapat mengakomodasi *specificity* dan *sensitifity*. AUC mampu menjadi nilai yang mampu mangakomodasi kedua nilai tersebut apabila tidak selaras. Nilai AUC terbaik pada analisis LORENS *full training set* adalah saat menggunakan 4 ruang partisi dan *threshold* 0,5. Nilai akurasi LORENS 4 partisi *threshold* 0,5 adalah 76,4% dengan sebesar *sensitivity* 85,9% dan *specificity* 69%. LORENS dengan menggunakan 4 partisi dan *threshold* 0,5 telah mampu memprediksi data *testing* dengan ketepatan sebesar 76,4%. *Sensitifity* sebesar 85,9% menunjukan bahwa model yang terbentuk mampu memprediksi gen Alzheimer tepat kedalam kelas Aktual gen Alzheimer sebesar 85,9%. Hal tersebut menunjukan hanya sekitar 13,1% gen Alzhiemer yang diprediksi kedalam kelas gen normal. Sedangkan *specificity* sebesar 69% menunjukan bahwa hanya 69% gen normal yang tepat diklasifikasikan kdalam kelas actual gen normal, sisanya sebesar 31% prediksi gen normal yang masuk kedalam kelas gen Alzheimer. Nilai AUC sebesar 0,774 menunjukan bahwa model LORENS 4 partisi dan *threshold* 0,5 sudah cukup baik untuk mengklasifikasikan data *testing*.

4.8 Analisis LORENS *Cross Validation*

Metode evaluasi yang baik digunakan untuk masalah klasifikasi adalah *Cross Validation.* Metode ini memperlakukan semua pengamatan secara adil dalam hal penentuan *training set data* dan *testing set data*. Metode evaluasi *Cross Validation* diharapkan mampu meningkatkan performa klasifikasi. *Cross Validation* pada penelitian ini mengunakan 10-*folds* dan besar partisi yang berbeda, yaitu 1 partisi, 2 partisi, 3 partisi, 4 partisi

dan 5 partisi. *Threshold* pada penelitian ini menggunkan *threshold* 0,5 dan *threshold* optimal, sedangkan jumlah *ensemble* yang digunakan sebesar 10.

Dalam analisis ini, data akan dibagi menjadi 10 bagian dengan jumlah yang adil. Masing-masing bagian akan diperlakukan sebagai data *training* dan data *testing* pada tiap *fold*. Data di f*old* ke-1 diperlakukan sebagai data *testing*, sedangkan *fold* ke-2 sampai ke-10 akan diperlakukan sebagai data *training*. Data di *fold* ke-2 akan diperlakukan sebagai data testing dan *fold* sisanya akan diperlakukan sebagai data *training*, begitu seterusnya sampai dengan *fold* ke-10. Jumlah model yang terbentuk dengan menggunakan jumlah *ensemble* 10, 10 *fold* dan 5 partisi adalah 2800 model *binary logistic regression*.

Dalam analisis LORENS, *threshold* yang didapat pada masing-masing *fold* akan berbeda.

Tabel 4.22 *Threshold* Optimal untuk 2 Partisi

Fold	Threshold Optimal	*Fold*	Threshold Optimal
1	0,5219	6	0,5344
2	0,525	7	0,5156
3	0,5188	8	0,5313
4	0,5313	9	0,5109
5	0,5313	10	0,5327

Pada *fold* pertama, didapatkan *threshold* optimal 0,521875, *fold* ke-2 adalah 0,525 sampai dengan *fold* ke-10 adalah 0,5326087.

Tabel 4.23 *Threshold* Optimal untuk 3 Partisi

Fold	Threshold Optimal	Fold	Threshold Optimal
1	0,5312	6	0,5188
2	0,5219	7	0,5281
3	0,5250	8	0,5313
4	0,5250	9	0,5264
5	0,5281	10	0,5171

Pada pembagian data menjadi 3 partisi, *fold* pertama didapatkan *threshold* optimal 0,5312, *fold* ke-2 adalah 0,5229 sampai dengan *fold* ke-10 adalah 0,5171.

Tabel 4.24 *Threshold* Optimal untuk 4 Partisi

Fold	Threshold Optimal	Fold	Threshold Optimal
1	0,5188	6	0,5281
2	0,5219	7	0,5313
3	0,5219	8	0,5281
4	0,5316	9	0,5202
5	0,5281	10	0,5233

Pada pembagian data menjadi 4 partisi, *fold* pertama didapatkan *threshold* optimal 0,5188, *fold* ke-2 adalah 0,5219 sampai dengan *fold* ke-10 adalah 0,5233.

Tabel 4.25 *Threshold* Optimal untuk 5 Partisi

Fold	Threshold Optimal	Fold	Threshold Optimal
1	0,5281	6	0,5281
2	0,5281	7	0,5281
3	0,5281	8	0,5219
4	0,5188	9	0,5326
5	0,5188	10	0,5202

Sedangkan pada pembagian data menjadi 5 ruang partisi, *fold* pertama didapatkan *threshold* optimal 0,5281, *fold* ke-2 adalah 0,5281 sampai dengan *fold* ke-10 adalah 0,5202. Nilai probabilitas akhir dari masing-masing *ensemble* dan partisi akan dibandingkan dengan nilai tersebut. Apabila nilai probabilitasnya lebih dari *threshold* optimal yang diperoleh, maka gen akan diklasifikasikan sebagai gen Alzheimer, jika kurang dari *threshold* maka gen akan diklasifikasikan sebagai gen normal. Untuk menghitung nilai *accuracy*, *sensitifity* dan *spesifity*, perlu dilakukan perhitungan tabel tabulasi silang antara kelas actual dan kelas prediksi. Berikut adalah tabel tabulasi silang pada masing-masing partisi dan *threshold* yang digunakan.

Tabel 4.26 Tabulasi Silang Hasil Klasifikasi LORENS dengan *Cross Validation*

			Kelas Aktual			
			Threshold 0,5		Threshold Optimal	
			+	-	+	-
Kelas Prediksi	2 partisi	+	63	12	60	11
		-	35	68	38	69
	3 partisi	+	63	12	59	11
		-	35	68	39	69
	4 partisi	+	66	14	57	11
		-	32	66	41	69
	5 partisi	+	68	14	58	11
		-	30	66	40	69

Dari hasil klasifikasi diatas, dapat dihitung nilai *accuracy, sensitivity* dan *specifity* serta *Area Under Curve* (AUC). AUC sangat baik digunakan untuk memilih model maupun metode terbaik untuk masalah klasifikasi. *Accuracy* merupakan ukuran seberapa baik model yang terbentuk untuk meramalkan suatu data *testing*. *Sensitivity* merupakan ukuran kebaikan model yang berguna mengukur seberapa baik model yang terbentuk untuk memprediksi secara tepat data *testing* di kelas positif yang tepat terprediksi kedalam kelas positif. *Specificity* untuk mengukur seberapa baik model yang terbentuk untuk memprediksi kelas negatif tepat kedalam kelas negatif.

Tabel 4.27 Ukuran Kebaikan LORENS dengan *Cross Validation*

Partisi	Accuracy	Sensitifity	Specificity	AUC
2 Partisi Threshold 0,5	0,7359	0,850	0,6429	0,7464
2 Partisi Threshold Opt	0,7247	0,8625*	0,6122	0,7374
3 Partisi Threshold 0,5	0,7359	0,850	0,6429	0,7464
3 Partisi Threshold Opt	0,7191	0,8625*	0,6020	0,7323
4 Partisi Threshold 0,5	0,7416	0,825	0,6735	0,7492
4 Partisi Threshold Opt	0,7079	0,8625*	0,5816	0,7220
5 Partisi Threshold 0,5	0,7528*	0,825	0,6939*	0,7594*
5 Partisi Threshold Opt	0,7135	0,8625*	0,5918	0,7272

Catatan : * adalah nilai terbesar.

Tabel 4.27 diatas menunjukan bahwa ukuran akurasi terbaik analisis LORENS untuk klasifikasi gen yang terkait Alzheimer adalah 5 partisi dengan *threshold* 0,5, dengan nilai *accuracy* sebesar 75,28%. Nilai *sensitifity* yang dihasilkan oleh LORENS 5 partisi dan *threshold* 0,5 adalah 82,5%, tidak lebih tinggi daripada partisi lainnya. Hal tersebut berarti metode LORENS 5 partisi dengan *threshold* 0,5 hanya mampu memprediksi gen Alzheimer tepat kedalam kelasnya sebesar 82,5%. Selaras dengan nilai *accuracy*, nilai *specificity* LORENS 5 partisi dengan *threshold* 0,5 memberikan nilai yang paling baik diantara partisi lainnya, yaitu sebesar 75,94%. Hal tersebut berarti bahwa metode LORENS 5 partisi dengan *threshold* 0,5 mampu memprediksi gen normal tepat kedalam kelasnya sebesar 75,94%. Namun untuk memilih metode terbaik disarankan menggunakan AUC, karena nilai AUN mampu mengakomodasi kelemahan *sensitivity* dan *specificity* yang cenderung bertolak belakang apabila kelas respon yang digunakan tidak *balance*. Nilai AUC terbaik pada analisis LORENS dengan prosedur evaluasi 10-*folds Cross Validation* adalah pada saat menggunakan 5 partisi dan *threshold* 0,5.

4.9 Pemilihan Metode Terbaik

Pemilihan metode terbaik dapat dilakukan dengan membandingkan ukuran ketepatan klasifikasi yang didapatkan. Ukuran ketepatan klasifikasi yang baik untuk digunakan adalah *Area Under Curve* (AUC). AUC mampu menangani kelemahan *sensitifity* dan *specificity* yang cenderung tidak selaras apabila kelas respon *inbalance*. Analisis *Naïve Bayes Classiffier* akan dibandingkan dengan analisis *Binary Logistic Regression* serta analisis LORENS. Secara keseluruhan, LORENS dengan menggunakan *threshold* 0,5 cenderung menghasilkan ketepatan klasifikasi lebih baik dibandingkan dengan analisis LORENS dengan menggunakan *threshold* optimal, artinya kelas respon cenderung *balance*. perbandingan diakukan pada masing-masing prosedur evaluasi, yaitu *full training set* dan *10 folds Cross Validation.* Ketiga metode memiliki keunggulannya masing-

masing, sehingga perlu dicari metode terbaik dalam menangani kasus klasifikasi gen yang terkait dengan sindrom Alzheimer.

Tabel 4.28 Perbandingan Ketepatan Klasifikasi Pada Prosedur Evaluasi *Full Training Set*

Metode	*Acc*.	*Sens*.	*Spec*.	AUC
Naïve Bayes Classifier	71,35%	83,1%	63,55%	0,733
Binary Logistic Regression	79,8%*	69,4%	92,5%*	0,809*
LORENS 4 Part *Thres* 0,5	76,4%	85,9%*	69%	0,774

Catatan : * adalah nilai terbesar.

Pada data DNA *microarray* tentang gen yang terkait sindrom Alzheimer, analisis dengan menggunakan prosedur evaluasi *full training set* menghasilkan kesimpulan bahwa model *Binary Logistic Regression* mempunyai ketepatan klasifikasi yang lebih baik dibandingkan kedua metode lainnya. *Accuracy* yang dihasilkan oleh metode *Binary Logistic Reression* adalah 79,8% dan AUC sebesar 0,809, namun metode ini memiliki *sensitifity* yang terendah dibandingkan kedua metode lainnya. Dalam kasus ini, *sensitifity* yang tinggi sangat dibutuhkan dalam memprediksi gen Alzheimer kedalam kelas Alzheimer. Apabila *sensitifity* yang dihasilkan rendah, akan berakibat pada resiko kesalahan mengklasifikasikan gen Alzheimer kedalam kelas gen normal.

Dengan menggunakan prosedur evaluasi *Cross Validation*, dapat diperoleh metode terbaik untuk mengklasifikasikan gen yang terkait sindrom Alzheimer. Karena prosedur ini mampu menjadikan data menjadi data *testing* dan data *training* secara adil.

Tabel 4.29 Perbandingan Ketepatan Klasifikasi Pada Prosedur Evaluasi *Cross Validation*

Metode	*Acc*.	*Sens*.	*Spec*.	AUC
Naïve Bayes Classifier	55,06%	72,7%	47,15%	0,599
Binary Logistic Regression	65,2%	56,2%	78,1%*	0,671
LORENS 5 Part *Thres* 0,5	75,28%*	82,5%*	69,39%	0,759*

Catatan : * adalah nilai terbesar.

Berdasarkan tabel 4.29 diatas, dapat diketahui bahwa metode LORENS 5 partisi dengan *threshold* 0,5 mengasilkan nilai *accuracy, senitifity* dan AUC terbesar dibandingkan kedua metode lainnya. *Accuracy* yang dihasilkan LORENS jauh lebih baik dibandingkan kedua metode lainnya, begitu juga nilai AUCnya. Pada kasus ini, nilai *sensitifity* juga sangat penting untuk diperhatikan. Karena akan lebih baik apabila tingkat akurasi klasifikasi gen Alzheimer yang tepat terklasifikasi kedalam kelas Alzheimer lebih tinggi dari yang lain. Dengan prosedur evaluasi *Cross Validation* dan mempertimbangkan semua ukuran ketepatan klasifikasi, metode LORENS adalah metode terbaik dalam kasus ini. Metode LORENS menghasilkan *accuracy* sebesar 75,28% dan AUC sebesar 0,759. Dengan nilai tersebut, dapat disumpulkan metode ini sudah tergolong baik untuk mengklasifikasikan data.

(Halaman ini sengaja dikosongkan)

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil analisis dan pembahasan klasifikasi gen yang terkait sindrom Alzheimer menggunakan metode *Naïve Bayes Classifier* dan *Logistic Regression Ensemble* adalah sebagai berikut.

1. Klasifikasi menggunakan *Naïve Bayes Classifier m*enghasilkan ukuran *accuracy* dari hasil prediksi sebesar 0,71348 dengan *sensitivity* sebesar 0,83098 dan *specificity* 0,6355. Analisis menggunakan metode *Naïve Bayes Classifier* telah mampu memprediksi ekpresi gen Alzheimer dengan ketepatan sebesar 71,35%. Klasifikasi menggunakan prosedur *Naïve Bayes Classifier* dan evaluasi *Cross Validation* guna mengkonfirmasi apakah model yang terbentuk sudah baik mengklasifikasikan data *testing*. *Accuracy* yang dihasilkan dalam analisis ini adalah 55,06%, sedangkan *sensitivity* dan *specifity* sebesar 72,73% dan 47,15%. AUC yang dihasilkan adalah sebesar 59,94%.
2. Analisis *Binary Logistic Regression* menggunakan 8 variabel prediktor terbaik, menghasilkan *accuracy* 79,8% dan AUC sebesar 0,809. Namun metode ini hanya menghasilkan *sensitifity* sebesar 69,4%. Dengan demikian, metode ini kurang baik untuk mengklasifikasikan gen yang terkait sindrom Alzheimer. Analisis *Binary Logistic Regression* dengan menggunakan prosedur evaluasi *Cross Validation* menghasilkan *accuracy* yang tidak terlalu tinggi, yaitu sebesar 65,2%. AUC yang dihasilkan sebesar 0,671, sehingga metode ini kurang baik untuk dalam menangani kasus pada penelitian ini.
3. Analisis LORENS menggunakan *full training set* memberikan hasil klasifikasi terbaik menggunakan 4 partisi dan *threshold* 0,5. *Accuracy* yang dihasilkan pada analisis ini adalah 76,4%, yang berarti model yang terbentuk mampu

melakukan klasifikasi dengan tingkat ketepatan sebesar 76,4%. Nilai *sensitivity* dan *specificity* yang dihasilkan adalah sebesar 85,9% dan 69,0%. Artinya adalah sebanyak 85,9% gen Alzheimer mampu diklasifikasikan dengan tepat kedalam kelas gen Alzhemer dengan ketepatan 86,9%. Sedangkan nilai *specifity* sebesar 69,0% berarti bahwa prediksi klasifikasi gen normal dapat tepat terklasifikasi kedalam gen normal dengan ketepatan 69,0%. Klasifikasi menggunakan LORENS dengan prosedur evaluasi *Cross Validation* menghasilkan partisi terbaik sebesar 5 partisi dengan *threshold* 0,5. Ukuran kebaikan model yang dihasilkan beberapa diantaranya adalah, *accuracy* sebesar 0,752809, *sensitivity* sebesar 0,693878 dan *specifity* sebesar 0,825. Analisis menggunakan metode LORENS sudah cukup baik untuk mengatasi masalah klasifikasi gen yang diangkat dalam penelitian ini, karena telah mampu memprediksi secara tepat 75,28%.

4. Metode terbaik dipilih berdasarkan nilai AUC terbesar. Pada kasus klasifikasi gen yang terkait dengan sinrom Alzheimer, metode LORENS *Cross Validation* 5 partisi *threshold 0,5*. Dengan nilai AUC sebesar 0,774, LORENS 4 partisi dengan *threshold* 0,5 sudah cukup baik untuk mengatasi masalah klasifikasi pada data DNA *microarray*.

5.2 Saran

Saran yang dapat diberikan oleh penulis untuk penelitian selanjutnya antara lain adalah dalam menganalisis menggunakan metode LORENS, diperlukan lebih banyak variabel prediktor. Sifat LORENS yang mampu menangani variabel prediktor yang sangat besar, seharusnya dapat dimanfaatkan dengan baik dengan cara menggunakan variabel prediktor dengan jumlah yang lebih besar. Sehingga informasi untuk melakukan analisis klasifikasi bertambah dan diharapkan dapat meningkatkan akurasi dari klasifikasi yang dilakukan.

DAFTAR PUSTAKA

Ambica, A., Gandi, S., & Kothalanka, A. (2013). An Efficient Expert System For Diabetes by Naive Bayesian Classifier. *International Journal of Engineering Trends and Technology (IJETT) - Volume 4 Issue 10 (2013)*: 4634-4639.

Anonim. (2015, Augustus 27). *DNA Microarray Technology*. Diambil kembali dari https://www.genome.gov/10000533/dna-microarray-technology/.

Asfihani, A. (2014). Prediksi Pembelotan Konsumen Software Antivirus 'X' dengan Binary Logistic Regression dan Logistic Regression Ensembles. *Tugas Akhir ITS*.

Association, A. (2016).*https://www.alz.org/documents_ custom/2016-facts-and-figures.pdf.* Diambil kembali dari https://www.alz.org:

Bekkar, M., Djemaa, H.K., Alitouche, T.A. (2013). Evaluation Measures for Models Assesment Over Imbalanced Data Sets. *Journal of Information Engginering and Application*, Vol. 3, N0.10.

Catal, C. (2012). Performance Evaluation Metrics for Software Fault Prediction Studies. *Acta Polytechnica Hungarica*, Vol. 9, N0.4.

Chou, S., Shan, J., Guo, Y. & Zhang, L., 2010. Automated Breast Cancer Detection and Classification Using Ultrasound Image : A Survei, *Pattern Recognition*. Volume 43, pp. 299-317.

Cowell, J., & Hawthorn, L. (2007). The Application of Microarray Technology to the Analysis of the Cancer Genome. *Current Molecular Medicine 7(1):103-20.*

Erke, A. R. V. & Pattynama, P. M. T., 1998. Receiver operating characteristic (ROC) analysis: Basic principlesn\ and application in radiologi. *European Journal of Radiology*, pp. 88-94.

Gorunescu, F. (2010). *Data Mining : Concepts, Models and Techniques.* Berlin: Springer.

Hosmer, D.W., Lemeshow, S. (2010). *Applied Logistic Regression, Second Edition.*. New York: John Wiley & Sons, Inc.

Kuswanto, H., Asfihani, A., Sarumaha, Y., & Ohwada, H. (2015). Logistic Regression Ensemble for Predicting Customer Defection with Very Large Sample Size. *Procedia Computer Science 72 , 86-93*.

Lee, K., Ahn, H., Moon, H., Kodell, R., & Chen, J. (2013). Multinomial Logistic Regression Ensembles. *Biopharm Stat*, 23(3):681-94.

Lim, N. (2007). Classification by Ensembles from Random Partitions using Logistics Models. *Stony Brook University*.

Lim, N., Ahn, H., Moon, H., & Chen, J. (2010). Classification of high-dimensional data with ensemble of logistic regression models. *Journal of Biopharmaceutical Statistics 20*, 160-171.

Lin, M., Lucas, H., & Shmueli, G. (2013). Too Big to Fail: Large Samples and The P-Value Problem. *INFORM Vol. 24 Issue 4, PP 1-12.*

Matsumoto, A., Aoki, S., & Ohwada, H. (2015). Comparison of Random Forest and SVM for Raw Data in Drug Discovery: Prediction of Radiation Protection and Toxicity Case Study. *Tokyo University of Science*.

Medhekar, D., Bote, M., & Deshmukh, S. (2013). Heart Disease Prediction System using Naive Bayes.

International Journal of Enhanced Reasearch in Science Technology & Engineering Vol 2 Issue 3,

Moorthy, K., & Mohamad, M. S. (2011). Random Forest for Gene Selection and Microarray Data Classification. *Bioinformation, Vol. 7 Issue 3, pp. 142-146.*

Nishiwaki, K., Kanamori, K., & Ohwada, H. (2015). Finding a Disease-Related Gene from Microarray Data using Random Forest. *Tokyo University of Science*.

Witten, I., Frank, E., & Hall, M. (2011). *Data Mining : Practical Machine Learning Tools and Techniques 3rd Edition.* Burlington: Morgan Kaufmann.

Zakharov, R., & Dupont, P. (2011). Ensemble logistic regression for feature selection. In *IAPR International Conference on Pattern Recognition in Bioinformatics* (pp. 133-144). Springer Berlin Heidelberg.

Zweig, M. H. & Campbell, G., 1993. Receiver Operating Characteristic (ROC) Plots : A Fundamental Evaluation Clinical Medicine. *Clinical Chemistry* pp. 561-577.

(Halaman ini sengaja dikosongkan)

Lampiran 1. Data *Microarray* Ekspresi Gen

Observasi	Gen	X1	X2	X3	…	X20
		WWOX	TAGLN3	COL5A2	…	MT1H
1	0	0.10455	0.16266	0.36261	…	0.101382
2	0	0.12569	0.16398	0.33492	…	0.200145
3	0	0.11770	0.31747	0.04254	…	0.089123
4	0	0.18944	0.14008	0.28824	…	0.146996
5	0	0.17150	0.19875	0.20066	…	0.177114
6	0	0.12342	0.35576	0.25174	…	0.105056
7	0	0.18290	0.14605	0.26325	…	0.142603
8	0	0.16636	0.13724	0.35421	…	0.176364
9	0	0.14501	0.17636	0.25749	…	0.18738
10	1	0.21922	0.14633	0.03775	…	0.222294
11	1	0.17705	0.15556	0.10722	…	0.143395
12	1	0.14255	0.22143	0.17056	…	0.173444
13	1	0.20534	0.12385	0.20528	…	0.204963
14	1	0.17707	0.16019	0.14937	…	0.127548
15	1	0.11915	0.13328	0.14037	…	0.158335
⋮	⋮	⋮	⋮	⋮	…	⋮
170	0	0.11271	0.11190	0.11523	…	0.11013
171	0	0.11239	0.11248	0.11278	…	0.111918
172	0	0.11187	0.11236	0.11297	…	0.114033
173	0	0.11226	0.11443	0.11232	…	0.114348
174	0	0.11284	0.11049	0.11302	…	0.11461
175	0	0.11099	0.11398	0.10690	…	0.112826
176	0	0.11266	0.11386	0.11239	…	0.112266
177	0	0.11042	0.11375	0.11429	…	0.111668
178	0	0.10906	0.11306	0.11241	…	0.117574

Lampiran 2. Rata-Rata Variabel Prediktor Tiap Kelas

Gen	Rata-Rata		Standar Deviasi	
	K=0	K=1	K=0	K=1
WWOX	0.1381028	0.170704	0.046477	0.06795
TAGLN3	0.1592328	0.147026	0.076584	0.063395
COL5A2	0.1672606	0.123943	0.092901	0.076968
MT1F	0.1200998	0.168648	0.051962	0.093359
ERF	0.1210645	0.172003	0.055521	0.084011
APLNR	0.1005678	0.158551	0.060265	0.121859
WIF1	0.1587169	0.125922	0.095537	0.085538
GFAP	0.1129493	0.171914	0.047592	0.096546
ITPKB	0.1266403	0.167925	0.07535	0.072152
COLEC12	0.1176748	0.162504	0.055297	0.10467
LDHA	0.1570207	0.142645	0.074565	0.078271
SLC16A5	0.1317459	0.16768	0.065714	0.072857
NRN1	0.1576684	0.143964	0.074911	0.074393
SYT5	0.1465296	0.150477	0.084861	0.071391
VCAN	0.1350187	0.168736	0.062682	0.067423
NPTX2	0.1620328	0.131967	0.099761	0.064754
HPCA	0.1548598	0.134809	0.096658	0.076144
RAB6A	0.1567493	0.15188	0.067853	0.065017
WWTR1	0.1231821	0.16783	0.069974	0.08116
MT1H	0.1229359	0.16809	0.046395	0.093719

Lampiran 3. Peluang Posterior *Naïve Bayes Full Training Set*

No	Kelas 0 (Gen Normal)	Kelas 1 (Gen Alzheimer)
1	1.18293E-30	9.21668E-43
2	3.42712E-30	4.73178E-37
3	2.43908E-21	5.20569E-25
4	0.487301162	1.09315E-10
5	1.48516E-05	1.0251E-09
6	3.86025E-15	1.10613E-26
7	4.48719E-07	4.144E-08
8	3.78346E-12	1.40405E-11
9	8.07558E-07	0.00031873
10	6.81262E-23	5.73818E-16
11	3.76711E-11	7.55003E-06
12	9.01526E-09	1.73806E-15
13	1.55625E-09	1.43476E-06
14	0.00459972	0.005303563
15	9.26694E-10	0.001741873
⋮	⋮	⋮
170	0.002123503	7.58815E-06
171	0.002040281	9.93146E-06
172	0.001380159	2.25656E-06
173	0.001992087	3.34231E-06
174	0.002425658	5.10171E-06
175	0.003227214	1.59346E-05
176	0.003468847	8.67686E-06
177	0.003316885	9.21795E-06
178	0.002203375	7.03717E-06

Lampiran 4. Peluang Posterior *Naïve Bayes Cross Validation*

No	Kelas 0 (Gen Normal)	Kelas 1 (Gen Alzheimer)
1	9.30978E-22	1.33021E-24
2	3.00974E-14	5.28713E-12
3	4.06898E-11	5.87824E-07
4	0.01088504	8.31031E-10
5	3.04372E-23	7.2863E-19
6	1.47887E-50	1.10791E-48
7	6.11943E-36	2.51973E-17
8	1.59469E-35	1.21199E-17
9	3.3826E-08	6.6064E-09
10	0.000930149	5.99039E-06
11	0.000635243	1.04249E-06
12	0.000644555	2.70157E-06
13	0.002012512	2.37111E-05
14	0.001545823	6.03795E-06
15	0.002936709	5.98709E-06
⋮	⋮	⋮
170	0.000403369	6.51871E-06
171	0.00011909	4.72962E-06
172	0.002327288	1.33896E-05
173	0.001539586	6.65434E-06
174	0.002207349	1.53418E-05
175	0.003341526	6.8317E-06
176	0.002128585	2.06337E-06
177	0.000603985	2.2832E-06
178	0.002113044	6.80462E-06

Lampiran 5. *Output Learning Decision* LORENS *Full Training Set*

Obs	2 part	2 part opt	3 part	3 part opt	4 part	4 part opt	5 part	5 part opt
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0
13	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
171	0	0	0	0	0	0	0	0
172	0	0	0	0	0	0	0	0
173	0	0	0	0	0	0	0	0
174	0	0	0	0	0	0	0	0
175	0	0	0	0	0	0	0	0
176	0	0	0	0	0	0	0	0
177	0	0	0	0	0	0	0	0
178	0	0	0	0	0	0	0	0

Lampiran 6. Alokasi Variabel Prediktor Pada LORENS 2 Partisi *Threshold* 0,5

Variabel	***Ensemble***									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
X1	2	1	2	1	1	2	2	1	2	2
X2	1	2	2	2	1	1	2	1	2	2
X3	2	2	2	2	2	2	1	1	2	1
X4	2	2	2	1	2	2	1	2	1	1
X5	2	1	1	2	1	1	2	2	1	2
X6	2	2	2	2	1	1	1	2	1	1
X7	1	1	2	1	1	2	2	1	1	2
X8	1	1	1	2	2	1	2	2	1	1
X9	2	1	1	1	2	1	2	1	2	2
X10	2	1	1	1	2	1	1	2	2	1
X11	1	1	2	2	2	1	2	1	1	2
X12	1	1	1	1	2	2	2	1	1	1
X13	2	2	1	2	1	2	1	2	1	2
X14	2	2	1	1	1	2	1	1	2	1
X15	1	1	2	1	1	2	2	2	2	1
X16	1	2	2	2	2	1	1	1	2	2
X17	1	1	1	2	2	1	1	2	2	1
X18	1	2	1	1	1	2	1	2	2	2
X19	1	2	1	1	1	2	2	1	1	2
X20	2	2	2	2	2	1	1	2	1	1

Lampiran 7. Alokasi Variabel Prediktor Pada LORENS 2 Partisi *Threshold* Optimal

Variabel	Ensemble									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
X1	2	2	2	1	1	1	1	2	1	1
X2	1	2	2	2	2	2	2	2	1	2
X3	1	2	1	1	2	2	2	2	2	2
X4	1	1	1	1	1	1	1	1	2	1
X5	1	2	2	1	1	1	2	2	1	2
X6	1	2	2	2	1	1	2	1	2	1
X7	2	1	1	1	2	1	2	1	1	1
X8	2	2	1	2	1	1	1	1	2	1
X9	1	1	2	1	1	1	2	1	2	2
X10	1	1	1	1	1	2	1	2	2	2
X11	2	1	1	1	2	2	2	1	2	2
X12	2	1	1	2	2	1	1	2	2	1
X13	2	1	2	2	1	2	2	1	1	2
X14	2	1	1	2	2	1	1	2	1	1
X15	1	2	1	2	2	2	1	2	2	2
X16	2	2	2	1	1	1	1	2	1	2
X17	2	2	1	1	2	2	2	1	2	1
X18	1	1	2	2	1	2	1	1	1	1
X19	2	1	2	2	2	2	2	2	1	1
X20	1	2	2	2	2	2	1	1	1	2

Lampiran 8. Alokasi Variabel Prediktor Pada LORENS 3 Partisi *Threshold* 0,5

Variabel	*Ensemble*									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
X1	1	1	3	2	3	1	2	3	3	2
X2	2	3	1	2	3	2	2	2	1	1
X3	1	2	3	1	1	2	3	1	2	1
X4	1	3	1	1	2	2	1	3	2	3
X5	2	3	3	3	2	1	1	3	3	1
X6	3	1	2	2	3	1	3	2	1	3
X7	3	3	3	1	1	2	3	1	3	2
X8	1	3	2	3	3	3	3	3	2	1
X9	1	3	3	1	3	1	2	2	2	3
X10	2	2	2	2	2	1	3	2	2	2
X11	2	2	2	3	1	1	3	1	1	3
X12	2	1	2	1	2	3	1	3	1	2
X13	3	2	1	2	3	3	2	1	3	3
X14	3	1	3	1	2	3	3	2	3	3
X15	3	2	1	3	1	2	2	1	1	2
X16	3	3	1	3	1	3	1	3	3	1
X17	1	2	1	1	2	2	1	3	2	3
X18	1	1	2	3	1	3	1	1	3	1
X19	2	1	1	3	1	3	1	1	1	1
X20	3	1	3	2	3	1	2	2	1	2

Lampiran 9. Alokasi Variabel Prediktor Pada LORENS 3 Partisi *Threshold* Optimal

Variabel	***Ensemble***									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
X1	1	3	3	1	3	1	1	1	1	2
X2	1	3	1	2	1	1	3	2	1	3
X3	1	1	1	1	1	2	3	3	2	1
X4	3	2	1	3	2	3	2	3	2	3
X5	2	1	3	3	2	1	1	2	1	3
X6	1	2	2	3	3	2	3	2	2	2
X7	2	1	3	1	1	2	2	3	3	1
X8	1	1	3	3	3	3	2	1	3	1
X9	3	2	3	2	2	1	1	1	3	1
X10	3	3	1	1	1	3	1	3	1	3
X11	3	1	2	3	2	1	1	1	2	3
X12	3	2	1	2	3	3	2	1	3	3
X13	1	1	1	1	3	1	3	3	1	2
X14	3	3	2	2	2	2	1	2	1	1
X15	2	2	2	3	3	1	2	2	1	3
X16	1	1	3	2	1	2	2	3	2	1
X17	2	3	3	3	3	3	3	1	3	2
X18	2	3	1	1	1	3	3	2	3	2
X19	2	2	2	2	1	3	3	3	3	1
X20	3	3	2	1	2	2	1	1	2	2

Lampiran 10. Alokasi Variabel Prediktor Pada LORENS 4 Partisi *Threshold* 0,5

Variabel	Ensemble									
	1	2	3	4	5	6	7	8	9	10
X1	1	3	1	2	2	2	2	3	4	2
X2	4	3	3	4	3	4	3	4	4	1
X3	4	1	2	4	2	1	3	3	2	2
X4	1	2	1	3	1	2	4	3	4	4
X5	2	1	4	1	3	4	4	3	2	3
X6	1	4	4	1	4	1	3	4	1	1
X7	2	3	3	3	3	1	3	3	2	1
X8	1	1	3	4	4	3	2	2	3	4
X9	3	3	4	4	3	4	3	4	1	3
X10	4	4	2	3	1	3	1	1	4	1
X11	1	1	1	3	4	4	1	1	3	2
X12	4	3	4	2	4	3	1	4	1	3
X13	2	2	3	2	2	3	2	2	3	2
X14	3	2	1	1	2	2	4	1	1	1
X15	2	4	2	3	2	4	2	4	2	2
X16	3	4	2	1	3	2	2	1	3	4
X17	3	1	1	2	4	2	4	1	1	3
X18	3	2	2	1	1	1	1	2	3	4
X19	2	4	4	2	1	3	4	2	2	3
X20	4	2	3	4	1	1	1	2	4	4

Lampiran 11. Alokasi Variabel Prediktor Pada LORENS 4 Partisi *Threshold* Optimal

Variabel	Ensemble									
	1	2	3	4	5	6	7	8	9	10
X1	3	3	4	1	3	4	2	3	1	1
X2	1	3	2	1	3	2	2	1	4	1
X3	2	4	3	3	1	1	3	2	4	4
X4	4	1	3	4	3	3	2	3	4	4
X5	1	2	2	2	4	3	4	2	1	1
X6	3	1	3	2	2	4	1	2	4	1
X7	3	3	3	1	1	1	3	4	2	1
X8	4	2	2	4	2	4	1	1	1	4
X9	2	2	4	3	4	2	4	4	4	4
X10	1	1	4	4	4	1	2	2	2	4
X11	2	3	1	3	1	2	4	2	2	3
X12	4	1	3	4	4	2	3	1	3	2
X13	1	4	4	4	2	1	2	4	1	2
X14	1	3	1	1	2	3	4	3	3	3
X15	3	2	4	2	3	4	3	4	3	3
X16	4	4	1	2	2	2	4	1	1	3
X17	2	2	1	2	1	4	1	3	3	2
X18	4	1	2	1	1	1	3	1	2	2
X19	3	4	1	3	4	3	1	3	2	2
X20	2	4	2	3	3	3	1	4	3	3

Lampiran 12. Alokasi Variabel Prediktor Pada LORENS 5 Partisi *Threshold* Optimal

Variabel	Ensemble									
	1	2	3	4	5	6	7	8	9	10
X1	1	5	5	5	2	5	5	4	5	4
X2	5	2	2	2	3	5	1	2	4	3
X3	3	4	1	4	2	4	1	1	2	4
X4	4	4	1	1	4	4	5	5	5	4
X5	5	1	4	5	5	5	4	2	1	3
X6	2	4	3	2	5	3	5	3	2	2
X7	5	5	5	5	1	3	1	3	5	5
X8	4	3	2	4	3	2	4	1	4	5
X9	1	3	3	3	2	2	4	5	4	1
X10	5	3	1	1	5	4	1	5	2	2
X11	1	1	4	3	2	4	3	5	3	1
X12	3	4	2	2	3	2	4	4	1	1
X13	1	3	2	5	1	3	5	4	5	1
X14	2	5	1	1	3	1	2	1	3	3
X15	3	5	5	4	1	3	3	3	1	2
X16	4	1	5	3	4	2	3	2	3	5
X17	4	2	4	1	5	1	2	1	1	4
X18	3	2	4	2	4	1	2	4	4	2
X19	2	2	3	3	1	5	2	2	3	5
X20	2	1	3	4	4	1	3	3	2	3

Lampiran 13. Koefisien Model Regresi Logistik LORENS 2 Partisi *Threshold* 0,5

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-1.9	-3.1	-3.3	-2.6	-2.4	-3.1	-2.6	-1.2	-3.5	-3.6
Partisi ke-2	-3.4	-2.0	-1.6	-3.1	-3.2	-1.8	-2.5	-3.8	-1.4	-1.9
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	-4.3	1.3	12.4	0.4	4.7	3.0	6.3	8.2	9.3	6.9
X2	-4.9	-3.2	-2.3	-0.3	-6.7	-0.2	-5.3	-4.9	-5.9	-10.2
X3	-11.3	-12.8	-8.4	-10.3	-11.1	-9.8	-12.3	-9.4	-10.8	-10.9
X4	8.4	5.7	4.3	11.8	5.5	10.8	7.7	8.5	2.6	8.1
X5	11.4	8.3	9.6	10.5	12.6	10.4	8.6	11.5	9.9	11.4
X6	5.8	2.2	2.9	6.7	6.3	4.0	1.8	3.3	7.2	2.6
X7	-8.0	-7.8	-5.5	-7.5	-7.9	-3.9	-6.9	-4.9	-6.7	-9.7
X8	17.6	18.0	16.9	4.7	12.1	8.3	12.5	14.3	6.1	15.3
X9	-3.7	3.1	-9.1	5.9	-6.8	-5.9	-0.1	3.4	2.3	0.3
X10	6.5	10.5	4.2	9.7	10.4	4.8	8.2	6.8	4.0	12.2
X11	-2.7	7.1	5.7	5.5	3.5	3.1	6.1	6.9	4.9	1.2
X12	8.4	3.6	4.5	3.7	10.0	4.1	1.6	4.2	1.9	8.1
X13	1.1	-0.3	-7.3	2.0	-3.7	-2.6	2.1	-4.8	-3.5	-2.2
X14	1.9	6.2	0.4	-0.1	-0.9	3.5	6.1	4.2	5.8	3.0
X15	-9.0	-13.6	-1.6	-11.8	-1.7	-2.9	-9.5	-18.3	5.4	-15.7
X16	-7.7	-10.7	-8.2	-7.5	-9.2	-6.4	-11.4	-5.5	-7.3	-7.5
X17	-4.9	-5.2	-5.5	0.1	-0.8	-3.4	-2.5	-6.1	-0.9	-2.4
X18	20.1	10.5	8.9	3.3	13.6	4.9	6.8	9.1	9.3	18.5
X19	6.3	5.2	3.3	5.3	2.6	7.7	6.8	7.6	-3.2	5.2
X20	12.0	13.9	14.3	13.5	12.8	10.3	14.9	6.3	8.5	9.8

Lampiran 14. Koefisien Model Regresi Logistik LORENS 2 Partisi *Threshold* Optimal

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-3.4	-2.4	-3.4	-2.7	-3.4	-2.7	-3.5	-1.8	-2.1	-2.6
Partisi ke-2	-1.9	-2.9	-2.9	-2.8	-2.0	-2.0	-2.3	-1.9	-3.3	-2.4
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	3.6	4.2	-1.6	-4.1	-4.5	-0.1	-0.9	3.9	3.8	-0.2
X2	-4.2	2.0	-2.8	-2.4	-4.5	1.0	-3.1	-0.6	-10.2	-2.1
X3	-11.0	-9.6	-9.2	-8.4	-10.4	-10.9	-8.2	-9.8	-9.7	-10.0
X4	9.7	9.8	13.1	13.4	8.7	7.2	6.3	3.1	12.3	5.8
X5	12.1	8.8	11.6	13.4	10.9	8.9	14.1	7.8	10.9	9.3
X6	5.2	7.3	6.8	3.7	2.7	5.1	5.9	2.0	0.8	4.9
X7	-5.5	-6.9	-5.6	-6.5	-5.4	-4.0	-5.7	-7.2	-10.9	-6.2
X8	11.5	6.7	19.3	11.5	10.6	8.4	15.3	9.6	18.2	10.3
X9	-1.1	0.7	-3.1	2.3	-5.3	-3.0	2.0	-0.8	1.5	-0.9
X10	6.1	7.3	14.7	7.3	5.9	7.9	12.7	3.6	12.2	8.9
X11	4.3	-3.2	4.3	11.8	4.2	-9.1	7.9	-5.0	1.8	4.1
X12	5.6	3.7	6.0	9.7	8.7	3.2	10.0	3.4	9.1	5.4
X13	1.5	-1.1	-3.2	-6.7	-3.3	-0.8	0.0	-7.3	-2.1	4.5
X14	0.5	0.5	4.1	-1.6	3.5	-1.3	-5.0	1.9	-2.6	2.9
X15	-8.3	-5.2	-14.8	-12.0	-3.5	-8.0	-16.7	0.2	-16.5	-1.1
X16	-6.0	-4.8	-6.9	-6.0	-5.9	-2.2	-11.0	-3.2	-10.4	-9.8
X17	-3.7	0.8	-3.3	-2.0	-1.0	1.7	0.1	-1.4	-2.1	-4.7
X18	6.8	5.5	8.4	5.7	7.5	9.2	5.9	15.2	22.9	4.1
X19	2.9	2.4	0.4	3.9	7.3	8.3	5.4	8.5	1.6	-1.6
X20	12.4	13.7	13.9	11.0	17.7	17.4	12.3	6.8	12.9	16.9

Lampiran 15. Koefisien Model Regresi Logistik LORENS 3 Partisi *Threshold* 0,5

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-2.0	-2.7	-1.1	-1.7	-0.3	-3.3	-2.4	-0.5	-2.6	-2.4
Partisi ke-2	-2.5	-0.9	-2.8	-2.1	-2.8	-0.8	-1.6	-2.4	-2.3	-2.7
Partisi ke-3	-1.8	-2.3	-2.2	-2.7	-2.0	-2.3	-2.5	-2.4	-1.6	-2.1
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	8.7	-0.5	5.5	4.5	6.8	-3.5	8.4	-0.1	9.0	0.5
X2	-4.4	0.3	0.3	-1.7	-0.9	-0.8	-3.5	-3.3	-2.6	-2.2
X3	-9.9	-9.0	-9.2	-8.7	-7.6	-8.0	-9.0	-7.7	-10.0	-9.9
X4	6.5	8.7	10.2	12.0	9.1	11.0	9.1	7.4	8.9	8.9
X5	9.6	8.5	7.3	8.1	9.2	11.9	8.6	6.6	11.7	9.5
X6	6.7	7.9	2.6	4.4	3.0	8.0	1.8	5.0	7.7	6.3
X7	-6.0	-4.7	-5.1	-3.1	-5.5	-3.6	-4.9	-3.9	-7.1	-8.0
X8	10.6	9.4	11.4	13.2	8.5	12.4	15.5	8.6	12.0	11.4
X9	-0.9	-0.7	1.4	5.6	-2.1	-3.3	3.7	2.3	-1.8	3.7
X10	3.8	6.4	4.8	3.6	4.7	2.7	9.0	4.2	9.2	10.7
X11	1.6	-3.9	-4.9	-2.0	-4.7	-1.4	1.8	-4.4	-2.5	-1.2
X12	3.9	9.3	5.9	4.7	2.8	7.3	3.0	4.1	8.6	6.0
X13	1.7	2.8	2.2	-3.2	-4.8	-2.5	-3.0	-0.7	-2.2	-0.6
X14	3.3	-2.0	2.3	4.1	-1.3	-1.6	6.6	2.1	-4.9	4.2
X15	2.4	9.3	-0.7	-6.5	7.7	6.5	-2.0	5.1	-5.5	-4.9
X16	-7.0	-3.2	-7.3	-5.8	-8.7	-6.8	-7.5	-4.1	-7.0	-6.7
X17	0.3	2.3	-2.6	-1.2	-3.1	2.0	-5.1	-3.4	1.1	-3.6
X18	1.0	-3.2	2.7	7.4	15.4	6.3	7.2	8.3	12.2	11.9
X19	5.1	-1.2	7.3	6.8	6.3	3.1	3.2	8.1	1.7	5.4
X20	14.0	11.8	15.1	9.9	6.1	12.2	9.1	9.3	14.0	17.5

Lampiran 16. Koefisien Model Regresi Logistik LORENS 3 Partisi *Threshold* Optimal

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-1.7	-2.1	-2.1	-2.1	-0.5	-2.0	-2.8	-2.5	-2.1	-1.5
Partisi ke-2	-2.1	-2.7	-2.1	-1.3	-2.7	-1.7	-1.8	-2.6	-1.2	-1.9
Partisi ke-3	-2.6	-1.9	-2.0	-3.4	-2.4	-2.8	-0.9	-1.6	-2.0	-2.8
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	12.2	4.3	6.9	5.7	4.9	5.7	-1.4	2.0	4.9	5.4
X2	-0.4	-3.7	-3.7	-0.4	-1.7	-4.6	0.5	-2.7	-3.1	-5.1
X3	-8.8	-8.3	-10.6	-8.6	-6.7	-8.8	-8.5	-8.3	-8.3	-9.5
X4	4.2	8.8	12.2	9.5	7.6	5.3	8.1	13.5	6.0	11.1
X5	12.3	10.7	7.2	12.9	11.1	10.1	10.2	12.7	10.4	10.2
X6	2.1	6.6	7.0	6.6	5.0	7.4	4.8	8.0	4.6	5.7
X7	-5.9	-5.3	-4.6	-7.8	-6.5	-4.1	-4.4	-5.9	-6.1	-3.3
X8	11.5	15.1	12.2	11.1	14.5	10.7	10.8	10.8	13.6	14.0
X9	-1.0	4.5	-1.7	3.6	0.7	0.8	-0.9	-2.6	-1.7	-2.5
X10	6.7	5.7	7.4	7.6	6.1	5.8	6.4	9.5	4.5	6.4
X11	-4.1	9.2	-4.6	3.9	-0.5	3.2	-1.6	-2.5	2.5	2.0
X12	6.7	6.1	8.0	6.8	7.4	6.4	6.6	7.9	6.9	2.3
X13	3.5	-0.2	1.2	-2.9	-0.9	-2.8	0.6	10.9	0.1	-4.3
X14	-1.7	2.6	2.7	-0.5	-2.6	7.0	-1.6	-1.4	-1.7	7.8
X15	-0.9	-6.2	-2.6	-10.9	-7.6	3.0	-1.4	0.5	1.2	-4.3
X16	-5.9	-4.1	-2.3	-5.7	-7.4	-5.3	-4.8	-9.2	-7.7	-3.1
X17	-3.3	-3.1	-2.0	-5.2	-3.7	-3.9	1.7	-4.2	-3.8	-1.6
X18	5.7	-0.7	2.8	6.8	12.0	-0.6	0.2	3.6	3.9	0.8
X19	8.0	2.2	3.6	6.9	9.6	-1.4	9.5	3.6	3.1	9.7
X20	9.7	10.5	11.4	16.6	5.1	19.1	10.7	8.1	13.8	9.8

Lampiran 17. Koefisien Model Regresi Logistik LORENS 4 Partisi *Threshold* 0,5

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-2.1	-2.2	-1.5	-2.3	-2.2	-1.5	-2.6	-0.4	-2.0	-1.2
Partisi ke-2	-1.8	-1.5	-0.6	-1.6	-1.2	-1.2	-1.8	-1.9	-1.5	-0.9
Partisi ke-3	-1.0	-1.2	-1.3	-1.7	-1.1	-2.6	-1.1	-1.9	-1.4	-2.2
Partisi ke-4	-2.4	-1.1	-2.8	-1.5	-2.4	-2.0	-2.5	-2.0	-2.1	-1.9
Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	5.1	9.1	6.7	5.3	9.0	7.8	10.6	5.2	4.3	11.3
X2	-2.2	-3.9	0.1	0.7	0.8	-3.9	2.8	-2.1	-4.8	-0.4
X3	-10.6	-9.1	-8.4	-8.8	-9.7	-8.1	-6.1	-8.1	-8.6	-8.6
X4	4.6	6.9	10.0	12.7	5.6	10.6	9.0	12.1	5.5	4.4
X5	10.4	10.5	9.7	12.7	8.6	10.8	10.2	9.2	11.6	10.0
X6	3.5	3.6	8.0	7.7	4.5	7.0	7.2	7.5	7.8	7.7
X7	-4.7	-5.1	-5.5	-6.8	-4.3	-5.5	-3.1	-3.2	-2.6	-5.6
X8	7.9	13.6	10.1	9.1	11.1	12.6	13.2	9.8	14.3	9.1
X9	10.3	4.4	-0.4	3.2	6.9	1.6	9.2	4.1	4.0	-1.0
X10	6.9	3.4	4.9	9.7	6.8	5.2	6.7	8.4	5.8	6.5
X11	-3.6	3.9	-2.9	0.3	-0.7	1.7	-4.8	0.8	-6.3	-2.0
X12	9.2	4.5	3.1	5.6	7.6	6.0	6.0	7.0	6.9	3.8
X13	-0.4	-7.7	-2.1	-2.5	-2.0	-3.2	-1.4	-7.8	-4.1	0.3
X14	2.8	2.4	2.0	-2.7	4.0	1.8	0.4	2.8	0.5	3.3
X15	1.1	3.6	9.8	-1.4	7.8	5.3	-2.8	0.1	2.2	6.1
X16	-7.1	-5.0	-7.7	-4.6	-3.0	-6.5	-5.2	-4.9	-6.0	-10.2
X17	-3.8	-1.1	-3.6	-2.5	-3.2	-3.4	-3.8	-2.6	-2.6	-3.4
X18	6.1	2.5	7.0	5.6	-3.8	4.3	0.5	4.6	13.9	4.8
X19	7.8	4.7	2.2	6.8	2.5	0.7	4.0	2.7	9.9	7.7
X20	16.3	8.4	8.4	8.8	6.8	15.8	12.2	6.3	6.5	7.4

Lampiran 18. Koefisien Model Regresi Logistik LORENS 4 Partisi *Threshold* Optimal

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-2.0	-2.6	-0.8	-0.8	0.9	-0.3	-1.9	-2.2	-2.3	-2.2
Partisi ke-2	-1.1	-2.2	-2.9	-2.1	-1.8	-1.3	-1.8	-2.4	-1.2	-1.4
Partisi ke-3	-1.7	-0.8	-1.8	-1.2	-1.7	-2.9	-1.0	-1.9	-2.0	-1.1
Partisi ke-4	-2.5	-1.0	-1.7	-2.8	-2.7	-2.0	-1.7	-1.2	-1.5	-2.1

Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	9.5	15.7	6.4	15.3	9.3	9.2	5.9	2.8	5.6	5.1
X2	-2.0	-6.8	-4.6	-8.9	-6.5	-2.3	-3.6	-3.7	1.0	-0.8
X3	-9.1	-9.7	-7.6	-9.4	-6.4	-6.2	-8.0	-8.1	-8.1	-9.5
X4	6.9	8.6	11.1	5.1	6.2	6.5	9.2	8.6	9.8	8.9
X5	12.0	9.8	9.5	10.7	9.3	10.8	10.9	12.9	6.5	9.8
X6	6.2	5.7	5.5	7.2	4.8	4.0	4.0	7.9	4.4	8.4
X7	-5.8	-6.3	-2.1	-7.3	-3.5	-5.6	-4.0	-5.7	-6.8	-4.6
X8	10.7	12.8	8.4	10.3	11.9	13.5	7.9	13.9	11.1	11.9
X9	6.7	-2.4	5.9	2.0	-0.3	8.1	6.1	5.2	6.0	-1.9
X10	6.8	3.8	4.3	4.9	4.0	10.4	4.9	3.6	7.4	9.1
X11	-0.9	2.4	1.2	-1.4	-6.9	3.3	4.7	3.2	-3.3	-0.8
X12	6.4	6.2	8.1	5.6	2.7	7.1	7.4	7.4	9.1	7.2
X13	0.7	5.4	-2.1	-3.7	-1.0	1.1	-1.6	-1.5	-0.8	-2.8
X14	-1.5	1.0	3.9	1.2	3.5	-2.5	-2.5	2.8	-0.8	1.7
X15	-0.4	1.4	-1.3	3.0	-1.2	-7.0	9.5	1.3	0.0	5.5
X16	-8.3	-9.7	-5.8	-2.5	-3.9	-6.1	-5.9	-7.2	-4.3	-9.1
X17	0.9	-3.7	-3.3	-1.3	0.1	-3.0	-2.7	-4.7	-4.9	-2.8
X18	3.6	-2.8	3.2	6.1	11.3	4.2	2.9	7.1	5.6	1.4
X19	4.2	8.0	11.1	7.1	5.4	3.5	0.7	5.6	7.7	8.4
X20	12.2	14.7	6.5	12.1	5.6	4.3	6.3	10.7	12.4	12.1

Lampiran 19. Koefisien Model Regresi Logistik LORENS 5 Partisi *Threshold* Optimal

Intercept	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
Partisi ke-1	-1.4	-2.1	-2.0	-2.0	-0.7	-1.1	0.0	-1.4	-2.0	-1.5
Partisi ke-2	-2.4	-0.7	-2.0	-1.7	-1.0	-1.9	-1.1	-1.7	-1.8	-1.4
Partisi ke-3	-1.1	-2.0	-2.3	-1.0	-2.1	-1.1	-0.9	-1.7	-0.8	-2.2
Partisi ke-4	-1.2	-1.9	-1.6	-1.3	-1.3	-1.3	-2.9	-1.4	-1.7	-1.4
Partisi ke-5	-1.6	-1.2	-0.8	-1.6	-2.3	-2.1	-1.9	-1.9	-1.5	-0.9

Variabel	ens1	ens2	ens3	ens4	ens5	ens6	ens7	ens8	ens9	ens10
X1	8.6	9.4	10.5	9.3	11.2	4.7	5.8	9.5	9.3	10.0
X2	-0.8	-1.2	-0.9	-2.5	-2.4	-3.8	1.5	-0.4	-2.2	-3.3
X3	-8.9	-8.7	-11.4	-8.7	-8.3	-9.0	-5.4	-9.3	-9.3	-9.7
X4	7.9	11.0	13.8	10.9	7.0	13.1	7.9	9.4	10.0	11.0
X5	10.6	9.4	13.1	8.6	11.5	9.6	7.1	9.9	9.3	11.1
X6	7.3	5.5	6.9	8.9	7.7	7.6	5.5	7.3	4.6	6.5
X7	-5.2	-6.2	-5.0	-5.0	-4.9	-5.2	-4.0	-5.9	-5.6	-4.6
X8	9.7	13.2	13.3	10.0	12.8	12.7	12.8	16.4	11.3	11.1
X9	5.6	-0.5	3.9	6.4	5.9	1.3	-0.9	3.4	3.9	7.7
X10	8.3	6.6	9.2	7.0	1.9	8.4	9.9	6.1	6.4	3.6
X11	-1.3	1.8	-1.3	1.4	-0.4	-0.5	-0.2	-3.0	-0.2	0.0
X12	7.9	8.3	7.2	8.1	8.4	7.0	3.6	5.1	3.9	6.1
X13	-2.2	-2.6	-3.1	-1.0	-1.0	0.7	-3.2	-4.0	-1.8	-1.8
X14	1.5	0.4	5.5	2.9	-1.7	4.3	3.9	7.6	2.2	-1.1
X15	9.4	5.6	6.7	0.6	4.5	6.7	5.3	1.0	5.6	4.4
X16	-5.4	-8.2	-5.7	-5.8	-11.3	-5.3	-9.2	-4.1	-5.8	-3.6
X17	-2.0	-1.6	-4.5	-4.6	-1.8	-5.8	-3.5	-2.5	-3.4	0.2
X18	0.6	-0.4	5.3	0.5	3.7	-1.4	-1.6	0.1	1.3	-1.8
X19	2.1	10.2	0.0	6.9	7.6	6.1	10.7	8.3	11.0	5.1
X20	8.9	13.5	8.2	9.3	11.0	12.7	12.1	12.1	14.3	10.9

Lampiran 20. *Syntax R* untuk *split data*.

```
alzheimer<-read.table("E:/datata.csv",header=TRUE)

#load the data

#Split the data frame
splitDataFrame<-function

(dataframe,seed=null,n=trainSize){
 if(!is.null(seed))set.seed(seed)
 index<-1:nrow(dataframe)
 trainindex<-sample(index,n)
 trainset<-dataframe[trainindex,]
 testset<-dataframe[-trainindex,]
 list(trainset=trainset,testset=testset)
 }
# Training Data 90% and Testing Data 10%
 split<-splitDataFrame(alzheimer,NULL,round(nrow

(alzheimer)*0.90))
 train90<-split$trainset
 test10<-split$testset
write.csv(train90, "E:\\train90.csv")
write.csv(test10, "E:\\test10.csv")
```

Lampiran 21. *Syntax R* untuk *Logistic Regression Ensemble*.

```
lr.cerp <- function(y,x,nens,fixsize=NULL,fixthres=NULL,search=F) {
        # initialization
        set.seed(as.numeric(Sys.time()))
        options(warn=-1)
        if(sum(is.na(x))>0) stop("missing value is found")
        if(sum(is.na(y))>0) stop("missing value is found")
        y <- as.data.frame(y)
        x <- as.data.frame(x)
        num_pred <- ncol(x)
        num_obs <- nrow(x)
        pos_rate <- sum(y)/num_obs

        # parameter search or default option
        if(search==T) {
                optimal <- search.thre_size(y,x,"lr")
                optsize <- optimal$size; opthreshold <- optimal$threshold
        }
        else {
                if(is.null(fixsize)) fixsize <- round(6*num_pred/num_obs)
                if(is.null(fixthres)) fixthres <- (pos_rate+.5)/2
                optsize <- fixsize; opthreshold <- fixthres
        }

        # main body
        ptss <- floor(seq(1,optsize+.999,length.out=num_pred))
        fitted <- NULL; predicted <- NULL; cname <- NULL; coef.table<-
matrix(0,num_pred,nens);
        partition.table <- matrix(0,num_pred,nens); intc <- matrix(0,optsize,nens);
probability <- rep(0,num_obs)
        for (i in 1:nens) {
                cname <- c(cname, paste("ens",i,sep=""))
                rand_pred <- sample(ptss)
                partition.table[,i] <- rand_pred
                avg_fit <- rep(0,num_obs)
for(j in 1:optsize) {
                        smp_dt <- cbind(y,x[,rand_pred==j])
                        intlr <- glm(y~.,data=smp_dt,family=binomial())
                        coef.vector <- intlr$coefficient
                        coef.vector[is.na(coef.vector)] <- 0
                        intc[j,i] <- coef.vector[1]; coef.vector <- coef.vector[-1]
                        coef.table[rand_pred==j,i] <- coef.vector
                        avg_fit <- avg_fit + intlr$fitted.values
                }
                fitted <- cbind(fitted,avg_fit/optsize,deparse.level=0)
                probability <- probability+(avg_fit/optsize)/nens
        }
        learning.decision <- ens.voting(fitted,opthreshold)$final.vote
        colnames(fitted) <- cname
        colnames(intc) <- cname
        colnames(coef.table) <- cname; rownames(coef.table) <- colnames(x)
        colnames(partition.table) <- cname; rownames(partition.table) <- colnames(x)

        return(list
```

```
(fitted=fitted,probability=probability,learning.decision=learning.decision,
                        partition.table=partition.table,coef.table=coef.table,intercept=intc,

        number.ensemble=nens,optimal.size=optsize,optimal.threshold=opthreshold))
}

### lr.cerp.predict applies lr.cerp model to new data(test set) similar as predict.lm function.
### lr.cerp.object is required and built from lr.cerp function.
### xtest is also required and should be same format as x in lr.cerp function.
### ytest is optional if you want to check the accuracy
lr.cerp.predict <- function(lr.cerp.object,xtest,ytest=NULL) {
# initialization
        options(warn=-1)
        if(sum(is.na(xtest))>0) stop("missing value is found")
        if(sum(is.na(ytest))>0) stop("missing value is found")
        xtest <- as.data.frame(xtest)
        num_obs <- nrow(xtest)
        nens <- lr.cerp.object$number.ensemble
        optsize <- lr.cerp.object$optimal.size
        opthreshold <- lr.cerp.object$optimal.threshold

        # main body
        cname <- NULL; test.decision <- NULL; fitted <- NULL; probability <-
rep(0,num_obs)
        xtest <- xtest[,rownames(lr.cerp.object$partition.table)]
        for (i in 1:nens) {
                avg_fit <- rep(0,num_obs)
                cname <- c(cname, paste("ens",i,sep=""))
                curmod <- lr.cerp.object$partition.table[,i]
                for(j in 1:optsize) {
                        intc <- lr.cerp.object$intercept[j,i]
                        wrkmat <- xtest[,curmod==j]
                        cvec <- lr.cerp.object$coef.table[curmod==j,i]
                        int_vl <- as.matrix(wrkmat)%*%cvec
                        int_vl <- int_vl + intc
                        int_vl[int_vl>=709] <- 709
                        avg_fit <- avg_fit + exp(int_vl)/(1+exp(int_vl))
                }
                fitted <- cbind(fitted,avg_fit/optsize,deparse.level=0)
                probability <- probability+(avg_fit/optsize)/nens
        }
        test.decision <- ens.voting(fitted,opthreshold,ytest)
        colnames(fitted) <- cname

return(list(fitted=fitted,probability=t(probability),decision=test.decision$final.vote,
optimal.size=optsize,optimal.threshold=opthreshold,decision.table=test.decision$twobytwo))
}

### lr.cerp.cv performs v-fold cross-validation using lr.cerp and lr.cerp.predict functions.
### Options and requirements are the same as lr.cerp function.
### One additional requirement is v_fold which is the number of fold to be performed for cross-
validation.
```

```
lr.cerp <- function(y,x,nens,fixsize=NULL,fixthres=NULL,search=F) {
        # initialization
        set.seed(as.numeric(Sys.time()))
        options(warn=-1)
        if(sum(is.na(x))>0) stop("missing value is found")
        if(sum(is.na(y))>0) stop("missing value is found")
        y <- as.data.frame(y)
        x <- as.data.frame(x)
        num_pred <- ncol(x)
        num_obs <- nrow(x)
        pos_rate <- sum(y)/num_obs

        # parameter search or default option
        if(search==T) {
                optimal <- search.thre_size(y,x,"lr")
                optsize <- optimal$size; opthreshold <- optimal$threshold
        }
        else {
                if(is.null(fixsize)) fixsize <- round(6*num_pred/num_obs)
                if(is.null(fixthres)) fixthres <- (pos_rate+.5)/2
                optsize <- fixsize; opthreshold <- fixthres
        }

        # main body
        ptss <- floor(seq(1,optsize+.999,length.out=num_pred))
        fitted <- NULL; predicted <- NULL; cname <- NULL; coef.table<-
matrix(0,num_pred,nens);
        partition.table <- matrix(0,num_pred,nens); intc <- matrix(0,optsize,nens);
probability <- rep(0,num_obs)
        for (i in 1:nens) {
                cname <- c(cname, paste("ens",i,sep=""))
                rand_pred <- sample(ptss)
                partition.table[,i] <- rand_pred
                avg_fit <- rep(0,num_obs)
for(j in 1:optsize) {
                        smp_dt <- cbind(y,x[,rand_pred==j])
                        intlr <- glm(y~.,data=smp_dt,family=binomial())
                        coef.vector <- intlr$coefficient
                        coef.vector[is.na(coef.vector)] <- 0
                        intc[j,i] <- coef.vector[1]; coef.vector <- coef.vector[-1]
                        coef.table[rand_pred==j,i] <- coef.vector
                        avg_fit <- avg_fit + intlr$fitted.values
                }
                fitted <- cbind(fitted,avg_fit/optsize,deparse.level=0)
                probability <- probability+(avg_fit/optsize)/nens
        }
        learning.decision <- ens.voting(fitted,opthreshold)$final.vote
        colnames(fitted) <- cname
        colnames(intc) <- cname
        colnames(coef.table) <- cname; rownames(coef.table) <- colnames(x)
        colnames(partition.table) <- cname; rownames(partition.table) <- colnames(x)

        return(list
```

```
lr.cerp.cv <- function(y,x,nens,v_fold,fixsize=NULL,fixthres=NULL,search=F) {
        # initialization
        set.seed(as.numeric(Sys.time()))
        options(warn=-1)
        if(sum(is.na(x))>0) stop("missing value is found")
        if(sum(is.na(y))>0) stop("missing value is found")
        y <- as.data.frame(y)
        x <- as.data.frame(x)
        num_obs <- nrow(y)
        rand_obs <- sample(1:num_obs)
        obs_rem <- num_obs%%v_fold
        obs_div <- (num_obs-obs_rem)/v_fold

        # main body
        probability <- rep(0,num_obs); predicted <- rep(0,num_obs); tbttable <-
matrix(0,2,2)
        part_size.list<-NULL; threshold.list<-NULL
        for(i in 1:v_fold) {
                if(i<=obs_rem) {head1<-(i-1)*(obs_div+1)+1;tail1<-i*(obs_div+1);}
                else {head1<-(i-1)*obs_div+obs_rem+1;tail1<-i*obs_div+obs_rem;}
                test_seq<-rand_obs[head1:tail1]
                learn_seq<-rand_obs[-c(head1:tail1)]
                ylearn<-y[learn_seq,];xlearn<-x[learn_seq,];xtest<-x[test_seq,];ytest<-
y[test_seq,]
                mid_rs<-lr.cerp(ylearn,xlearn,nens,fixsize,fixthres,search)
                pred_rs<-lr.cerp.predict(mid_rs,xtest,ytest)
                predicted[test_seq]<-pred_rs$decision
                for(j in 1:nens) probability[test_seq]<-
probability[test_seq]+pred_rs$fitted[,j]/nens
                tbttable<-tbttable+pred_rs$decision.table
                part_size.list<-c(part_size.list,mid_rs$optimal.size)
                threshold.list<-c(threshold.list,mid_rs$optimal.threshold)
        }

        return(
list(probability=probability,predicted=predicted,partition.size.list=part_size.list,
                threshold.list=threshold.list,decision.table=tbttable))
}

### internal functions
ens.voting <- function (tot_res,threshold,y=NULL) {
        nens<-ncol(tot_res);nobs<-nrow(tot_res)
        if (!is.null(y)) {real_pos<-sum(y);real_neg<-nobs-real_pos}
        tot_res[tot_res>=threshold] <- 1; tot_res[tot_res<threshold] <- 0
        final.vote <- rep(0,nobs)
        for(i in 1:nobs) final.vote[i] <- mean(tot_res[i,])
        final.vote[final.vote>=0.5] <- 1; final.vote[final.vote<0.5] <- 0
        twobytwo <- NULL
        if (!is.null(y)) {
                real_pred_pos <- sum(final.vote==y&y==1)
                real_pred_neg <- sum(final.vote==y&y==0)
                real_pos_pred_neg <- real_pos - real_pred_pos
                real_neg_pred_pos <- real_neg - real_pred_neg
                twobytwo <-
rbind(c(real_pred_pos,real_pos_pred_neg),c(real_neg_pred_pos,real_pred_neg))
```

```
rownames(twobytwo) <- c("real.pos","real.neg")
                        colnames(twobytwo) <- c("pred.pos","pred.neg")
        }
        return(list(final.vote=final.vote,twobytwo=twobytwo))
}
search.thre_size <- function (y,x,method) {
        nprd <- ncol(x);nobs <- nrow(x);orate <- sum(y)/nobs
        szseq <- NULL; int_fits <- NULL
        initseed <- c(2,3,4,5,6,7,8,9,10,12)
        for (i in initseed) {
                ipts<-i*nprd/nobs
                ipts<-floor(ipts)
                if (ipts%%2==0) ipts<-ipts+1
                if (szseq[length(szseq)]!=ipts||is.null(szseq)) {
                        szseq <- c(szseq,ipts)
                        int_fits <- cbind(int_fits,cv.fit(y,x,ipts,method))
                }
        }
        nsrsz <- length(szseq)
        add_fits<-NULL;addsz<-NULL
        if(orate>=.5) iseq<-seq(.5,orate,.02)
        else {iseq<-seq(.5,orate,-.02); iseq<-rev(iseq)}
        nbis<-length(iseq)
        szfth<-rep(0,nbis);acfth<-rep(0,nbis)
        for(j in 1:nbis) {
                acseq<-rep(0,nsrsz)
                for(k in 1:nsrsz) {
                        tmpf<-rep(0,nobs)
                        tmpf[int_fits[,k]>=iseq[j]]<-1;tmpf[int_fits[,k]<iseq[j]]<-0
                        acseq[k]<-sum(tmpf==y)/nobs
                }
                nbst<-sum(acseq==max(acseq));scol<-seq(1:nsrsz)
                if(nbst==1) nthc<-scol[acseq==max(acseq)]
                else {
                        tmpcol<-scol[acseq==max(acseq)]
                        nthc<-tmpcol[round(nbst/2)]
                }
                if(nthc==1) {
                        upts<-szseq[nthc+1];lpts<-szseq[nthc]
                        utfac<-acseq[nthc+1];ltfac<-acseq[nthc]
                        while(lpts!=upts) {
                                mpts<-(lpts+upts)/2
                                mpts<-floor(mpts)
                                if(mpts%%2==0) mpts<-mpts+1
                                if(mpts==upts) break
                                if(length(addsz)==0) {
                                        mtf<-cv.fit(y,x,mpts,method)
                                        addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                }
else if(sum(addsz==mpts)==0) {
                                        mtf<-cv.fit(y,x,mpts,method)
                                        addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                }
```

```
else mtf<-add_fits[,addsz==mpts]
                                        tmtf<-rep(0,nobs)
                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                        mtfac<-sum(tmpf==y)/nobs
                                        if(ltfac>utfac) {
                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                        else if(ltfac<utfac) {
                                                if(mtfac>=ltfac) {lpts<-
mpts;ltfac<-mtfac}
                                                else {lpts<-upts;ltfac<-utfac}
                                        }
                                        else {
                                                if(mtfac>=ltfac) {
                                                        lpts<-mpts;ltfac<-
mtfac
                                                        upts<-mpts;utfac<-
mtfac
                                                }
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                }
                                if(ltfac>=utfac) {szfth[j]<-lpts;acfth[j]<-ltfac}
                                else {szfth[j]<-upts;acfth[j]<-utfac}
                        }
                        else if(nthc==nsrsz) {
                                lpts<-szseq[nthc-1];upts<-szseq[nthc]
                                ltfac<-acseq[nthc-1];utfac<-acseq[nthc]
                                while(lpts!=upts) {
                                        mpts<-(lpts+upts)/2
                                        mpts<-floor(mpts)
                                        if(mpts%%2==0) mpts<-mpts+1
                                        if(mpts==upts) break
                                        if(length(addsz)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else if(sum(addsz==mpts)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else mtf<-add_fits[,addsz==mpts]
                                        tmtf<-rep(0,nobs)
                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                        mtfac<-sum(tmpf==y)/nobs
                                        if(ltfac>utfac) {
                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
```

```
else mtf<-add_fits[,addsz==mpts]
                                        tmtf<-rep(0,nobs)
                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                        mtfac<-sum(tmpf==y)/nobs
                                        if(ltfac>utfac) {
                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                        else if(ltfac<utfac) {
                                                if(mtfac>=ltfac) {lpts<-
mpts;ltfac<-mtfac}
                                                else {lpts<-upts;ltfac<-utfac}
                                        }
                                        else {
                                                if(mtfac>=ltfac) {
                                                        lpts<-mpts;ltfac<-
mtfac
                                                        upts<-mpts;utfac<-
mtfac
                                                }
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                }
                                if(ltfac>=utfac) {szfth[j]<-lpts;acfth[j]<-ltfac}
                                else {szfth[j]<-upts;acfth[j]<-utfac}
                        }
                        else if(nthc==nsrsz) {
                                lpts<-szseq[nthc-1];upts<-szseq[nthc]
                                ltfac<-acseq[nthc-1];utfac<-acseq[nthc]
                                while(lpts!=upts) {
                                        mpts<-(lpts+upts)/2
                                        mpts<-floor(mpts)
                                        if(mpts%%2==0) mpts<-mpts+1
                                        if(mpts==upts) break
                                        if(length(addsz)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else if(sum(addsz==mpts)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else mtf<-add_fits[,addsz==mpts]
                                        tmtf<-rep(0,nobs)
                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                        mtfac<-sum(tmpf==y)/nobs
                                        if(ltfac>utfac) {
                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                        else if(ltfac<utfac) {
```

```
if(mtfac>=ltfac) {lpts<-mpts;ltfac<-mtfac}
                                                                else {lpts<-upts;ltfac<-utfac}
                                                        }
                                                        else {
                                                                if(mtfac>=ltfac) {
                                                                        lpts<-mpts;ltfac<-
mtfac
                                                                        upts<-mpts;utfac<-
mtfac
                                                                }
                                                                else {upts<-lpts;utfac<-ltfac}
                                                        }
                                                }
                                                if(ltfac>=utfac) {szfth[j]<-lpts;acfth[j]<-ltfac}
                                                else {szfth[j]<-upts;acfth[j]<-utfac}
                                        }
                                        else {
                                                lpts<-szseq[nthc-1];upts<-szseq[nthc]
                                                ltfac<-acseq[nthc-1];utfac<-acseq[nthc]
                                                while(lpts!=upts) {
                                                        mpts<-(lpts+upts)/2
                                                        mpts<-floor(mpts)
                                                        if(mpts%%2==0) mpts<-mpts+1
                                                        if(mpts==upts) break
                                                        if(length(addsz)==0) {
                                                                mtf<-cv.fit(y,x,mpts,method)
                                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                                        }
                                                        else if(sum(addsz==mpts)==0) {
                                                                mtf<-cv.fit(y,x,mpts,method)
                                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                                        }
                                                        else mtf<-add_fits[,addsz==mpts]
                                                        tmtf<-rep(0,nobs)
                                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                                        mtfac<-sum(tmpf==y)/nobs
                                                        if(ltfac>utfac) {
                                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                                else {upts<-lpts;utfac<-ltfac}
                                                        }
                if(mtfac>=ltfac) {lpts<-mpts;ltfac<-mtfac}
                                                                else {lpts<-upts;ltfac<-utfac}
                                                        }
                                                        else {
                                                                if(mtfac>=ltfac) {
                                                                        lpts<-mpts;ltfac<-
mtfac
                                                                        upts<-mpts;utfac<-
mtfac
                                                                }
```

```
else {upts<-lpts;utfac<-ltfac}
                                        }
                                }
                                if(ltfac>=utfac) {lsps<-lpts;lsbs<-ltfac}
                                else {lsps<-upts;lsbs<-utfac}
                                upts<-szseq[nthc+1];lpts<-szseq[nthc]
                                utfac<-acseq[nthc+1];ltfac<-acseq[nthc]
                                while(lpts!=upts) {
                                        mpts<-(lpts+upts)/2
                                        mpts<-floor(mpts)
                                        if(mpts%%2==0) mpts<-mpts+1
                                        if(mpts==upts) break
                                        if(length(addsz)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else if(sum(addsz==mpts)==0) {
                                                mtf<-cv.fit(y,x,mpts,method)
                                                addsz<-c(addsz,mpts);add_fits<-
cbind(add_fits,mtf)
                                        }
                                        else mtf<-add_fits[,addsz==mpts]
                                        tmtf<-rep(0,nobs)
                                        tmtf[mtf>=iseq[j]]<-1;tmtf[mtf<iseq[j]]<-0
                                        mtfac<-sum(tmpf==y)/nobs
                                        if(ltfac>utfac) {
                                                if(mtfac>=utfac) {upts<-
mpts;utfac<-mtfac}
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                        else if(ltfac<utfac) {
                                                if(mtfac>=ltfac) {lpts<-
mpts;ltfac<-mtfac}
                                                else {lpts<-upts;ltfac<-utfac}
                                        }
                                        else {
                                                if(mtfac>=ltfac) {
                                                        lpts<-mpts;ltfac<-
mtfac
                                                        upts<-mpts;utfac<-
mtfac
                                                }
                                                else {upts<-lpts;utfac<-ltfac}
                                        }
                                }
                                if(ltfac>=utfac) {usps<-lpts;usbs<-ltfac}
                                else {usps<-upts;usbs<-utfac}
                                if(lsbs>=usbs) {szfth[j]<-lsps;acfth[j]<-lsbs}
                                else {szfth[j]<-usps;acfth[j]<-usbs}
                        }
                }
```

```
fnbst<-sum(max(acfth)==acfth);fscol<-seq(1:nbis)
        if(fnbst==1) {
                finsz<-szfth[max(acfth)==acfth]
                finth<-iseq[max(acfth)==acfth]
        }
        else {
                ftmpcol<-fscol[max(acfth)==acfth]
                tgcol<-ftmpcol[round(fnbst/2)]
                finsz<-szfth[tgcol]
                finth<-iseq[tgcol]
        }
        return(list(size=finsz,threshold=finth))
}

cv.fit <- function (y,x,npt,method) {
        num_pred<-ncol(x)
        num_obs<-nrow(x)
        lfit<-rep(0,num_obs)
        nv=3
        if(method=="lr") lfit<-lr.cerp.cv(y,x,1,nv)$probability
        else if(method=="lrt") lfit<-lrt.cerp.cv(y,x,1,nv)$probability
        else if(method=="ct") lfit<-ct.cerp.cv(y,x,1,nv)$probability
        return(lfit)
}
```

BIODATA PENULIS

Penulis dengan nama lengkap Reynaldi Wisnu Werdhana, biasa dipanggil Aldi lahir di Bojonegoro pada tanggal 27 Oktober 1994. Penulis merupakan anak pertama dari dua bersaudara, dari pasangan Nugroho Suci Anjalmo dan Sri Kanti. Penulis menempuh pendidikan dari TK – SMA dari tahun 1999 – 2013. Setelah lulus dari SMA Negeri 2 Bojonegoro tahun 2013, penulis melanjutkan studi di Departemen Statistika ITS melalui jalur SBMPTN. Selama menjadi mahasiswa ITS, penulis aktif dalam organisasi yakni berkontribusi di Departemen Kesejahteraan Mahasiswa HIMASTA-ITS 2014/2015 sebagai staf dan IHMSI sebagai delegasi pada tahun kedua. Pada tahun ketiga penulis juga berkontribusi di HIMASTA-ITS 2015/2016 sebagai Wakil Ketua Himpunan dan IHMSI 2014/2018 sebagai Kepala Badan Usaha Pusat. Selain aktif dalam pengembangan *softskill*, dalam bidang akademis penulis pernah menjadi asisten dosen Teknik Sampling dan Survey, Analisis Data 1 dan Pengendalian Kualitas Statistika. Saat ini penulis mampu menyelesaikan Tugas Akhir yang berjudul "Klasifikasi Gen yang Terkait Sindrom Alzheimer Menggunakan Metode *Naïve Bayes Classifier*, *Binary Logistic Regression* dan *Logistic Regression Ensemble*". Demikian biodata penulis yang dapat disampaikan. Segala bentuk saran dan kritik yang membangun, serta apabila pembaca ingin berdiskusi lebih lanjut mengenai Tugas Akhir ini, maka pembaca dapat menghubungi penulis dengan mengirimkan email ke reynaldiwisnu27@gmail.com.